175 PTAS

88

mi computer

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VIII-Fascículo 88

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185. 1.°, 08021 Barcelona
Tel. (93) 200 19 02

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 028510
Impreso en España-Printed in Spain-Octubre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Cuatro diseños
Durante los últimos años han salido al mercado numerosas impresoras térmicas. Aunque algunas máquinas, como la Floyd 40 y la Alphacom 32, están destinadas específicamente a su empleo con el Sinclair Spectrum, hay otras máquinas, como la Epson P40 y la Brother HR-5, que están diseñadas para trabajar con una amplia gama de micros personales

Chris Stevens

Hot metal

Analizaremos cuatro impresoras para el Spectrum y enunciaremos los criterios que deben guiar al usuario al hacer su elección

Uno de los primeros periféricos cuya compra consideran los usuarios de un ordenador personal es la impresora, que les permite producir copias de los listados de sus programas. Al elegir una impresora los usuarios del Spectrum se encuentran con una serie de problemas. Por una parte, este ordenador carece de las interfaces convencionales que por lo general utilizan las impresoras y, por otra, la impresora de Sinclair Research, la ZX Printer, cuya producción se ha suspendido, se caracterizaba por su confusa impresión y la tendencia de las copias a perder nitidez con el paso del tiempo. Debido a que Sinclair Research ha decidido no mejorar su impresora, otros fabricantes han producido por su propia cuenta impresoras térmicas de costo reducido. En este capítulo vamos a examinar algunas de estas alternativas.

A la hora de comprar una impresora se han de tener presentes varias consideraciones básicas. La principal es, por supuesto, el precio, consideración que con frecuencia implica mucho más que limitarse a comparar etiquetas de precio. Puede haber, por ejemplo, costos ocultos que no se incluyen en el precio de la máquina. El fabricante de una impresora puede afirmar que ésta está diseñada para un Spectrum, cuando en realidad requiere un conector de interface RS232C. Por consiguiente, antes de poder utilizar la impresora, también habrá de adquirirse una ampliación Interface 1 (que conecta al Spectrum un conector RS232C).

Costos ocultos

Otro costo oculto es el papel para la impresora. Muchas impresoras sólo utilizan papel diseñado específicamente para ellas. Por ejemplo, las impresoras de tipo térmico como la propia ZX Printer requieren un papel especial sensible al calor. Por lo tanto, tras pagar por su impresora, usted se verá de hecho ligado a los caprichos del fabricante, y el papel puede ser difícil de conseguir o bien tener un precio excesivo.

Al adquirir una impresora es esencial, en consecuencia, asegurarse de que la tienda que vende el dispositivo posea siempre existencias regulares de papel y otros servicios, y que usted sepa cuánto es probable que le cuesten. También ha de recabar información sobre el servicio técnico. La maquinaria que posee partes móviles mecánicas, como el cabezal de impresión, es más susceptible de sufrir desperfectos que los componentes electrónicos y, por tanto, es preferible comprar la impresora en una empresa que posea un eficiente servicio técnico.

Por último hay que considerar el problema de la compatibilidad de software. Todas las impresoras

están programadas para responder a códigos (por lo general, caracteres ASCII) que les indican que han de llevar a cabo ciertas funciones, tales como un retorno de carro, la cantidad de caracteres por línea y la densidad de impresión. Aunque muchos de estos códigos son idénticos a los que utiliza el Spectrum, quizá no sea así para todas las máquinas.

Sinclair Research, como muchos otros fabricantes de micros personales, ha adaptado muchos de los códigos ASCII para su propio uso y, aunque el manual de la impresora indique que un código tiene un determinado significado, bien podría tener otro distinto para el Spectrum. Aquí la solución consiste en ver las facilidades de la impresora operando con este ordenador antes de comprarla.

Las mismas precauciones son válidas para el software que piense utilizar. No hay nada más frustrante que unos códigos de instrucciones que funcionen perfectamente bien desde BASIC pero que no se puedan emplear desde un determinado paquete para tratamiento de textos. En este caso, usted se vería en la nada envidiable situación de tener que guardar su copia en cinta o disco, desconectar y volver a leer la copia como archivo secuencial para poder imprimirla.

Habiendo enumerado varios criterios para la adquisición de una impresora para el Spectrum, examinemos ahora algunas de las impresoras térmicas económicas disponibles en el mercado. De las impresoras que vemos a la derecha, la más barata es la Alphacom 32. Es, asimismo, la máquina que guarda un mayor parecido con la ZX Printer. Las fuentes de tipos utilizadas son como las de ésta y, al igual que en el caso de la máquina de Sinclair, sólo puede imprimir un máximo de 32 columnas.

Por alguna razón, los fabricantes han optado por equipar a la Alphacom con su propia fuente de alimentación. La adición de esta caja extra y los cables correspondientes es innecesaria, en especial porque el bus de ampliación del Spectrum posee una fuente de 9 V, que sería suficiente para operar una impresora térmica.

La estructura de la Floyd 40, de Shiva Marketing, parece más bien endeble. La carcasa se dobla con bastante facilidad y el rodillo que retiene el rollo del papel no es más que un delgado tarugo de madera. A pesar de esto, la máquina constituye una notable mejora respecto a la Alphacom.

La potencia se toma del bus de ampliación del Spectrum y por ello no hay ninguna caja extra que instalar. La Floyd 40 también utiliza papel blanco, con lo cual la impresión en negro resulta mucho más legible. Sin embargo, lo que realmente diferencia a esta máquina de la Alphacom y la ZX Printer es el hecho de que acepta ciertos caracteres de control que pueden alterar la salida a la impresora.

Estos caracteres se envían a la impresora a través de una instrucción LPRINT normal seguida de comillas. El carácter de control se encierra entre signos de exclamación, indicando a la impresora que no lo imprima porque se trata de una instrucción. Por ejemplo, la línea LPRINT"¡H!" le ordenaría a la Floyd 40 que imprimiera en caracteres de doble altura. El formateado se anula impartiendo la misma instrucción una segunda vez.

Otras facilidades disponibles incluyen caracteres para gráficos, cursiva, anchura doble y formato invertido. El cabezal de impresión de cinco por siete agujas produce atractivas imágenes en modalidad

MODELO	Alphacom 32	Floyd 40	Epson P40	Brother HR-5
PRECIO*	£54,95	£69,95	£99,95	£149,95
COSTO PAPEL	£10,95 (3 rollos)	£2,50 (c/rollo;10 rollos,£12,50)	£8,65 (5 rollos)	£4,75 (c/rollo)
INTERFACE EMPLEADA	Bus de ampliación	Bus de ampliación	RS232 o en paralelo Centronics	RS232 o en paralelo Centronics

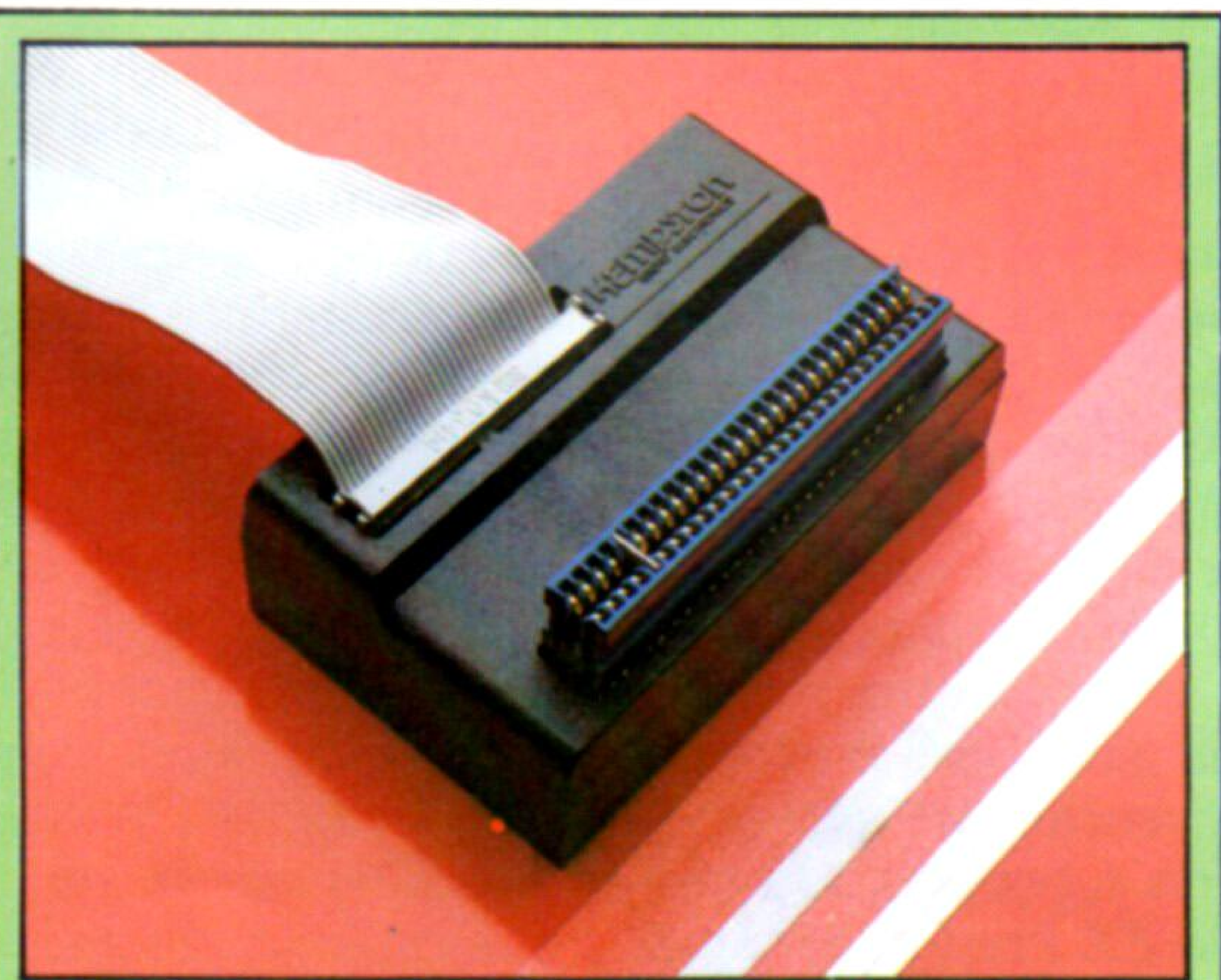
*En el mercado británico





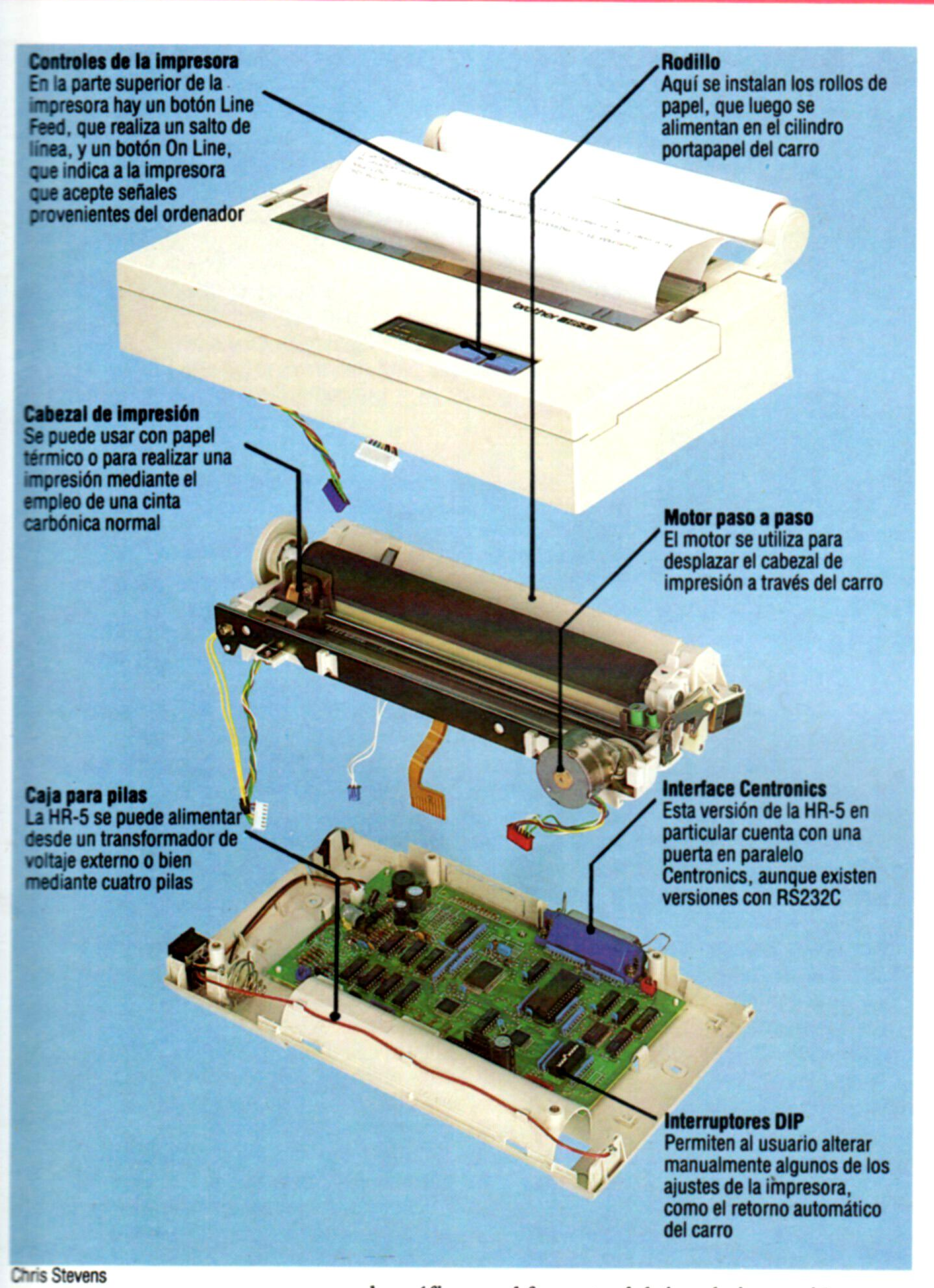






Interface Kempston
Para poder utilizar algunas de las impresoras más sofisticadas (equipadas con interfaces Centronics o RS232C), el usuario de un Spectrum deberá invertir en un dispositivo periférico adecuado que proporcione estas conexiones. Sin ninguna duda, de las interfaces de este tipo que existen en el mercado la más popular es la interface Kempston, que proporciona acceso a una amplia gama de impresoras compatibles con Centronics





Chris Stevens

de gráficos y el formato del tipo de impresión es tan satisfactorio como el de muchas otras máquinas que cuestan el doble. Aunque bastante lenta, el único serio inconveniente del sistema es que está limitado a un rollo de papel de 80 mm de ancho.

La tercera máquina que vemos es la Epson P40, que se vende en dos versiones básicas que comprenden los principales tipos de interface para impresora. La P40S es una versión en serie con un conector RS232C, mientras que la P40P posee en la parte posterior una puerta en paralelo Centronics. Sin embargo, puesto que el Spectrum no posee instaladas como estándar ninguna de las dos, es necesario adquirir una Interface 1 o bien una de las otras muchas disponibles en el mercado.

Surge el problema de que el conector RS232 que ofrece Sinclair en la Interface 1 es un conector D de siete patillas no estándar. Los manuales del P40 y la Interface 1 le indicarán de qué forma se han de posicionar las patillas y luego sólo resta una simple labor de soldadura. Sin embargo, si no siente especial inclinación por el uso de soldadores, quizá sea preferible comprar una de las interfaces producidas por empresas independientes de las que hay a la venta. Kempston, por ejemplo, proporciona interfaces que se enchufan en la parte trasera del Spectrum, así como conexiones estándares tanto para dispositivos RS232 como Centronics.

La Epson P40, a pesar de su tamaño relativamente pequeño, puede imprimir tanto en modalidad de 80 como de 40 columnas. Asimismo, la máquina puede hacer uso de una gran cantidad de códigos de escape de los que disponen los dispositivos más grandes de la gama. Por supuesto, el Spectrum no posee una tecla Escape *per se*, de modo que los códigos se deben enviar como códigos ASCII en el formato `CHR$(27);"E"`; (donde `CHR$(27)` representa el carácter Escape ASCII) que indica a la P40 que se ponga en modalidad enfatizada.

La P40 también puede llevar a cabo otras numerosas operaciones, tales como alterar el juego de caracteres, entrar en modalidad de imagen de bits (que permite que usted cree sus propios caracteres) y en modalidad condensada. Aparte de estas alteraciones, que se pueden realizar desde software, también hay interruptores DIP, que permiten establecer la paridad y la cantidad estándar de columnas de tipos a imprimir.

El hecho de que la P40 esté amparada bajo la prestigiosa marca comercial Epson hace que uno pueda sentirse bastante seguro de que contará con suficiente apoyo durante algún tiempo.

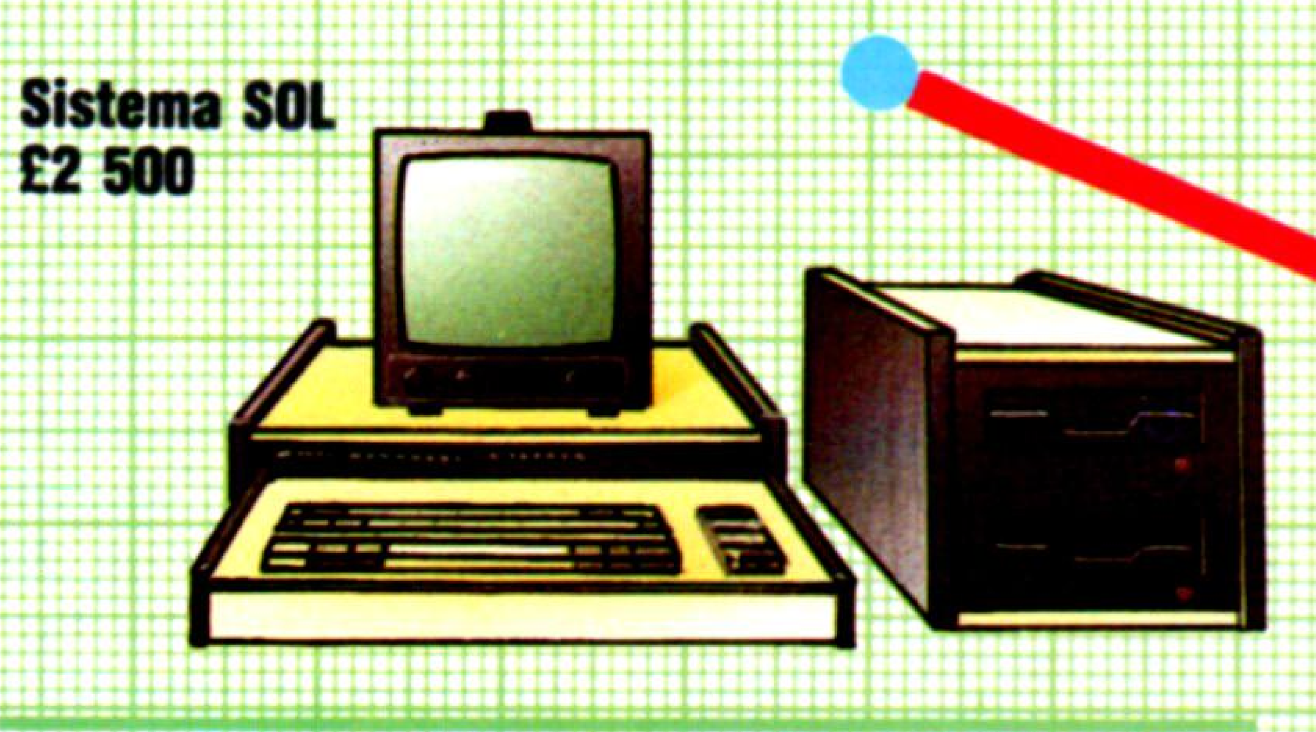
A diferencia de las otras impresoras que describimos en este capítulo, la Brother HR-5 puede usar papel térmico o bien, de estar equipada con una cinta, papel normal. Otra ventaja de esta máquina respecto a las otras es que el carro puede aceptar papel de tamaño A4 y, por tanto, se la puede utilizar para cartas y otras aplicaciones comunes de tratamiento de textos. La anchura adicional le permite imprimir hasta 132 caracteres por línea.

Al igual que la máquina Epson, la HR-5 hace un uso exhaustivo de códigos Escape para formatear su impresión y, por tanto, puede producir una amplia gama de caracteres, fuentes internacionales y otros formatos. Asimismo, es muy silenciosa en funcionamiento. Otra similitud que comparte la Brother HR-5 con la Epson es que, lamentablemente, la máquina viene con una interface en paralelo Centronics o bien con un conector RS232C. Nuevamente usted habrá, en consecuencia, de adquirir una interface adecuada para poder hacer funcionar la impresora desde un Spectrum. Aparte de está salvedad, la calidad de la salida impresa puede calificarse de excelente.

Al escoger una impresora se plantean problemas sólo si usted decide adquirir una máquina cuyas capacidades estén más allá de las de las máquinas construidas a la medida para el Spectrum. La Alphacom y la Floyd 40 ofrecen la ventaja de que sencillamente se enchufan en el ordenador y que aceptan instrucciones normales del Spectrum, tales como `COPY`, pero sus capacidades en realidad se limitan a los listados de programas.

Las impresoras situadas en el tramo superior de la escala de precios, con una gama más amplia de ampliaciones, están diseñadas para trabajar con máquinas muy diversas; por consiguiente, no están hechas a la medida para funcionar con el Spectrum. Sin embargo, hasta que alguien produzca una impresora de gran calidad diseñada específicamente para el Spectrum, el usuario habrá, lamentablemente, de adaptar la impresora al ordenador.

Control dinámico



Iniciamos una serie en la que examinaremos detalladamente el sistema operativo CP/M (Control Program for Microprocessors)

Cuando a comienzos de la década de los setenta se inventaron los sistemas de almacenamiento en disco flexible, obviamente existía una apremiante necesidad de desarrollar un sistema operativo que fuera capaz de organizar la información retenida en esos dispositivos sin que los usuarios tuvieran que hacerlo manualmente. Aproximadamente en la misma época, la firma norteamericana de microchips Intel decidió desarrollar un sistema de disco flexible para su serie 8000 de microprocesadores, que había introducido recientemente. El equipo de desarrollo de la empresa esta dirigido por un joven ingeniero asesor, Gary Kildall. El programa resultante se dio a conocer como CP/M (originalmente, siglas de *Control Program Monitor*, que posteriormente se cambiaría por *Control Program for Microprocessors*: programa de control para microprocesadores).

Cuando se lanzó el paquete, en 1975, se convirtió enseguida en un éxito rotundo. De hecho, su aceptación fue tal que cuando un grupo de diseñadores abandonó Intel para formar su propia empresa, Zilog Inc., decidieron que su nuevo chip (el Z80) fuera compatible con el 8008 de modo que pudiera ejectuar CP/M. El propio Kildall también se marchó de Intel para crear su propia empresa, Digital Research, que actualmente es una de las mayores empresas de software del mundo.

El éxito del CP/M se puede medir por el hecho de que, aunque durante los años setenta aparecieron muchos sistemas operativos rivales para micros de ocho bits, en especial para aquellos basados en el procesador 6502, no compatible con CP/M, ahora el sistema es el estándar industrial de facto para los micros de gestión de ocho bits. Esto resulta particularmente asombroso si se considera que el CP/M ya tiene más de un decenio de antigüedad, lo que, desde el punto de vista de la microinformática, es un extraordinario período de vida útil.

Para poder ejecutar CP/M se necesita un sistema de ordenador con un microprocesador 8008 o un chip compatible, una unidad de disco flexible y al menos 48 K de memoria disponible. Esta gran cantidad de RAM es necesaria porque con frecuencia se requerirá la memoria para retener el programa CP/M propiamente dicho y cualquiera de los archivos de instrucciones (que pueden ocupar hasta 8 K) y archivos en disco con los cuales puede estar operando. Con el advenimiento de los ordenadores de 16 bits, parece que la época dorada del CP/M en el mercado de gestión ha pasado ya. Pero el sistema ha sido adoptado por muchos fabricantes de ordenadores personales y educativos, posibilitando que sus micros Z80 se aparten del mercado de juegos y pasen a aplicaciones más "serias". La adición de una unidad de disco compatible con CP/M le proporciona instantáneamente al consumidor una inmensa cantidad de software que se desarrolló en los años setenta para micros de gestión basados en el 8008 y el Z80. Ello les permite a los fabricantes obviar el problema del apoyo de software, que ha perjudicado enormemente a muchos micros. Entre los fabricantes que en los últimos años han seguido este camino se incluyen Memotech, Research Machines y Amstrad.

A diferencia de la mayoría de ordenadores personales, que poseen sus sistemas operativos instalados "a bordo" (y que se activan con el encendido), el CP/M casi siempre reside en un disco flexible (denominado *disco de sistema*) que se carga desde una unidad de disco con el encendido. No existe ninguna razón especial por la cual el CP/M haya de estar retenido en un disco de sistema, puesto que algunas empresas han suministrado sistemas similares en ROM. Las causas son básicamente históricas. En los primeros ordenadores, la memoria era difícil de conseguir; de estar el CP/M instalado, cuando no se lo necesitara no hubiera hecho más que ocupar una preciosa memoria que se podría haber utilizado para otros fines. De este modo, se conservó en disco y se lo cargaba (LOAD) sólo cuando era necesario.

Típicamente, un disco de sistema CP/M contiene el programa CP/M propiamente dicho y varios programas más cortos a los que se puede llamar mediante el CP/M para que lleven a cabo funciones específicas de manipulación de disco. El CP/M es, sin embargo, más que un simple sistema de gestión: es un sistema operativo. Cuando se carga el CP/M es un ordenador con su propio sistema operativo residente en ROM, "anula" el sistema de la máquina y asume totalmente la operación del ordenador. Por lo tanto, cuando la máquina está operando bajo CP/M, éste no reconoce las instrucciones en BASIC que normalmente ejectua el ordenador, generando un mensaje de error. De hecho, un programa en BASIC no se ejecutará en absoluto bajo el CP/M a menos que el sistema operativo posea un intérprete o compilador de BASIC añadido al sistema.

£2 500
£2 250
£2 000
£1 750
£1 500
£1 250
£1 000
£750
£500
£250
£0



1975 1976 1977 1978 1979 1980

Kevin Jones

Tras el encendido, por lo general el disco de sistemas CP/M se ejecuta automáticamente y se visualiza (como podemos apreciar) un directorio con algunas de las instrucciones disponibles. Un atento examen del directorio del disco nos dice numerosas cosas sobre la forma en que está estructurado el CP/M. La serie de instrucciones (como todos los archivos en CP/M) consta de dos partes, empezando por el nombre "primario". Cuando se desea cargar cualquier archivo de instrucciones, simplemente se digita esta parte del nombre completo del archivo junto con el nombre del archivo en el cual deseamos operar, y se carga y ejecuta automáticamente.

La segunda parte del nombre de archivo completo (a la derecha del punto) se conoce como la extensión, que indica al CP/M (y al operador) qué clase de archivo es. Dado que todos los archivos de un disco de sistema CP/M son archivos de instrucciones, todos ellos llevan como sufijo la extensión COM. Más adelante en esta serie veremos otras clases de extensión de archivo.

Otra cosa que destaca cuando se examina el directorio del disco es que el programa CP/M propiamente dicho no se lista. A primera vista, entonces, puede parecer que el CP/M es meramente la suma de sus partes y un conjunto de programas de instrucciones más pequeños. Si bien, en gran medida, ésta es la forma en la que está estructurado el CP/M, un examen más atento del directorio revela que muchas instrucciones vitales (como LOAD) están presentes, mientras que faltan otras (p. ej., SAVE). Es evidente que el directorio no nos está contando toda la historia.

Hemos señalado anteriormente que un sistema operativo debe ser, en la medida de lo posible, transparente, dejando en manos del usuario el menor trabajo posible para administrar el sistema adecuadamente. Cuando cargamos desde el disco de sistemas, el CP/M se carga en la RAM para el usuario y allí permanece mientras el ordenador esté encendido. También se cargan junto con el programa CP/M principal un número de archivos de instrucciones utilizados comúnmente (llamados programas de utilidades) que también residen en RAM. Por consiguiente, para usar estas instrucciones no hay necesidad de cargarlas desde el disco.

Para evitar cualquier confusión sobre qué instrucciones ya están residentes en RAM y cuáles se han de cargar, se ha hecho que estas instrucciones, al igual que el programa CP/M propiamente dicho, sean "invisibles" para el directorio y se afirma que son instrucciones "incorporadas". Se dice que los archivos de instrucciones retenidos en disco son instrucciones "transitorias". Una vez ejecutadas, son desechadas por el sistema operativo y si usted desea volver a utilizarlas son cargadas de nuevo.

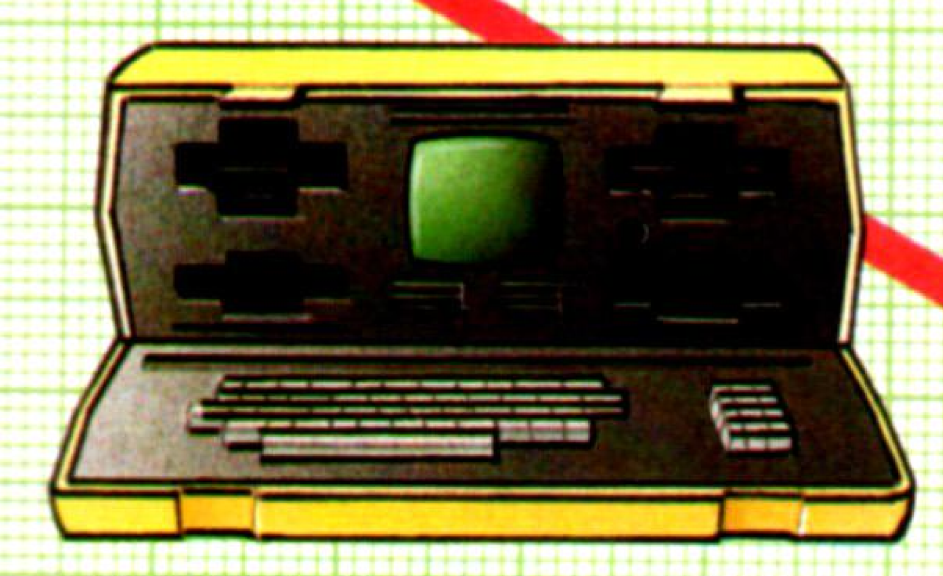
```
WRENMENU version 1.1
(c) 1984 Quantec Systems and Software Ltd
A>dir
A: WRENMENU COM : CPM3     SYS : COPYSYS  COM : ERASE    COM : SET      COM
A: SHOW     COM : SUBMIT   COM : TYPE     COM : DIR      COM : GET      COM
A: PUT      COM : RENAME   COM : SETDEF   COM : PIP      COM : HELP     COM
A: HELP     HLP : FC       COM : DEVICE   COM : FORMAT   SUB : FC       GET
A: FORMAT   GET : CONFIGUR COM
A>
```

Consulta al directorio
Digitando DIR se visualiza una lista de los archivos retenidos en el sistema CP/M. A menos que usted ya haya añadido algunos archivos en el disco, todas las entradas serán archivos de instrucciones (COMmand), identificables en función del sufijo .COM del final de cada nombre de archivo. El directorio visualiza, asimismo, otras informaciones, como la cantidad de memoria que ocupa cada archivo y el nombre del archivo desde el cual se accedió al directorio

Este sistema aparentemente arbitrario de ejecutar las instrucciones se debe, en parte, a limitaciones de hardware, y en parte también a que ello le confiere flexibilidad al sistema. Puesto que el CP/M se basa en procesadores de ocho bits, la máxima memoria que el procesador puede direccionar directamente son 64 K. Si en el encendido se cargaran en la memoria todas las instrucciones necesarias para ejecutar CP/M, quedaría muy poca RAM para programas del usuario y archivos de texto. Por consiguiente, se decidió que para optimizar el uso de la memoria central, ésta sólo contendría las instrucciones más habituales, mientras que el resto se cargarían al ser requeridas y se desecharían luego de ser utilizadas.

Sin embargo, tal sistema posee sus ventajas. Significa que se pueden añadir con bastante facilidad instrucciones transitorias extras con sólo añadirlas al directorio con una extensión .COM. Esta flexibilidad en cuanto a la posibilidad de añadir instrucciones extras que se puedan ejecutar bajo la cobertura general del CP/M ha tenido como consecuencia la inmensa cantidad de programas de aplicaciones que se han escrito para ejecutar bajo CP/M. A modo de ejemplo, el CP/M considera al popular paquete para tratamiento de textos *WordStar* (aunque se lo trata como un programa separado que se puede ejecutar bajo CP/M) como otro de sus archivos de instrucciones cuando el mismo se carga en el sistema. Esto reporta varias ventajas. Para empezar, el *WordStar* se puede usar en cualquier sistema CP/M: el programa no requiere rutinas propias de manipulación de disco, puesto que puede, simplemente, llamar a las instrucciones del CP/M que llevan a cabo estas funciones.

Osborne 1
£1 695

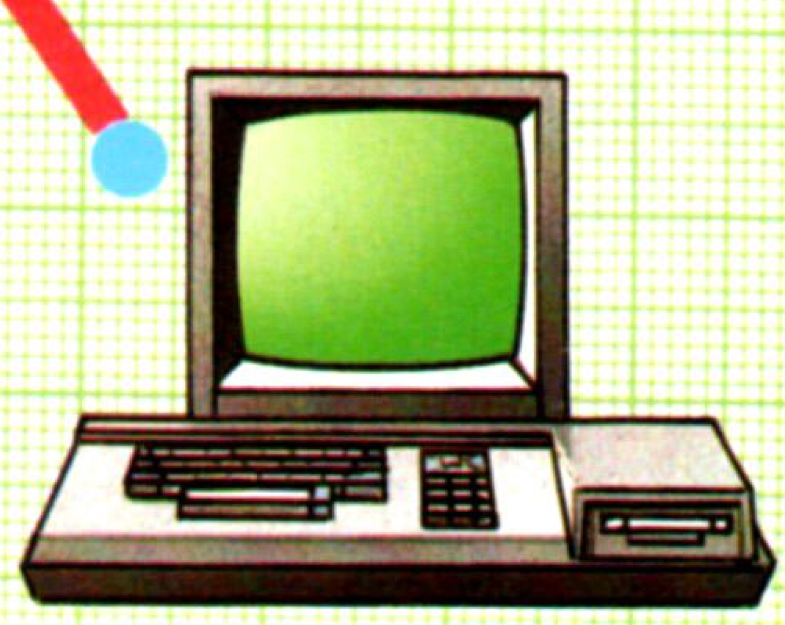


Wren
£1 000



(Los precios que aparecen en estas páginas corresponden al mercado británico.)

Amstrad CPC 664
£339



1981 **1982** **1983** **1984** **1985**

Provechoso intercambio

Ya en tierra, puede iniciar una interesante actividad de intercambio que, al regresar a casa, le reportará saneados beneficios

La línea 892 de la sección principal del programa llama a la rutina de intercambio, cuya primera tarea consiste en comprobar si usted ha traído algunas armas para comerciar. Aunque las armas pueden ser útiles (para defenderse de los piratas, etc.), al jefe de los nativos no le agradan las armas, porque sabe que éstas podrían ser fuente de problemas entre su pueblo. Esto significa, lamentablemente, que si *ha traído* armas para comerciar con ellas, el jefe rechazará la oferta y se visualizará un mensaje en ese sentido. No obstante, el resto de las mercancías se comerciará en lotes.

La rutina continúa y le informa cuál era el precio en su país, antes de que usted se hiciera a la mar, para cada una de las mercancías que ofrece el jefe. Primero se comercia la sal y usted puede intercambiarla por perlas, figurillas o especias. Tras comerciar con la sal, puede continuar con la tela de la misma manera, y luego con los cuchillos y las joyas.

Si no hay armas en el barco, no será necesario que el jefe le diga al capitán que no quiere armas, de modo que la línea 10072 examina el segundo elemento de la matriz de provisiones, OA(2), que registra la cantidad de armas, e ignora el mensaje del jefe si este elemento está establecido en cero.

La línea 10080 examina los cuatro últimos elementos de la matriz de provisiones (de OA(3) a OA(6)) en busca de mercancías aptas para el comercio. Los dos primeros elementos representan medicinas y armas, que no son aptas y, por consiguiente, no se verifican. En caso de que quedaran cuchillos, sal, tela o joyas, se le dirá que el jefe está deseoso de ofrecer a cambio perlas, figurillas y especias. Pero si todos los elementos de la matriz estuvieran establecidos en cero, el juego terminaría porque, como es obvio, no habría ninguna mercancía con la cual comerciar.

Tras informarle sobre el valor de cada una de las mercancías cuando el barco zarpó, y advirtiéndole que desde entonces los precios podrían haber experimentado variaciones, entre las líneas 10130 y 10200 el programa entra en un bucle para tratar la mecánica del intercambio. El código de este bucle calcula las cantidades de perlas, figurillas y especias ofrecidas por el jefe para cada lote de mercancías que usted desea comerciar, y le pregunta qué mercancías desea. Luego almacena las mercancías elegidas en el barco.

El bucle cuenta de 3 a 6, omitiendo, por consiguiente, las armas y los frascos de medicina (que están retenidos en el primer y segundo elemento de OA()) y se ocupa de cada artículo por separado. La línea 10135 comprueba si el valor del elemento de la matriz que se esté examinando en cada momento es 0, lo que significa que el jugador no posee ninguna mercancía de este tipo y, de ser así, pasa al siguiente elemento de OA() saltando a la instrucción NEXT de la línea 10200. En caso contrario, se informará al jugador la cantidad de cada mercancía, que se imprime mediante la línea 10145. Las líneas 10150-10153 imprimen el tipo de mercancía, correspondiente al valor actual de T.

Después de que el jefe ofrezca algunas perlas, figurillas y especias, el programa calcula las cantidades que recibirá usted de estos artículos por cada una de las mercancías que ofrezca. Para este cálculo se utiliza la matriz bidimensional EQ(,) creada en la línea 63 y que contiene las proporciones de trueque.

La expresión de la línea 10165 calcula la cantidad de perlas ofrecidas para cada elemento del bucle,



multiplicando la cantidad de sus mercancías a intercambiar, OA(T), por la proporción de trueque establecida en la matriz EQ(,). Puesto que el bucle está establecido de 3 a 6, se le debe restar 2 a T para equipararlo con la cantidad de elementos de las provisiones de EQ(,), que son de 1 a 4. Si el bucle se está ocupando de la sal, el contador del bucle T será igual a 3; EQ(T−2,1) corresponde, por consiguiente, a la intersección entre el primer elemento del primer subíndice, sal, con el primer elemento del segundo subíndice, perlas. En EQ(,), éste está establecido en .5; por lo tanto, para la sal, la línea 10165 multiplicará la cantidad de sal por .5 para calcular la cantidad de perlas que se ofrece para el trueque.

La línea 10166 lleva a cabo una función similar para la cantidad de figurillas que ofrece el jefe a cambio. La proporción de trueque para éstas está registrada en el segundo elemento del segundo subíndice de la matriz EQ(,). Cada valor sucesivo del contador del bucle T dirige el programa, por consiguiente, a la sección adecuada de la segunda columna de la matriz. Del mismo modo, la línea 10167 determina la cantidad de especias que ofrece el jefe por cada una de las mercancías de usted, utilizando la tercera columna de la matriz de proporciones de trueque.

Tras informársele de lo que ha ofrecido el jefe, usted debe decidir qué mercancías desea comerciar. Se debe entrar 1, 2 o 3 para el tipo de mercancías requerido, y la línea 10176 determina si la entrada fue correcta asegurándose de que la entrada esté comprendida en la escala de 1 a 3. Las mercancías intercambiadas se deben luego transportar hasta el barco; la línea 10180 almacena la cantidad en la matriz AO(), DIMensionada en la línea 68.

Los tres elementos de esta matriz representan las tres mercancías que tiene el jefe para ofrecer. La cantidad de artículos transportados al barco se vuelve a calcular utilizando la matriz de proporciones de trueque EQ(,), multiplicando el elemento en cuestión por la cantidad de mercancías comerciadas, OA(T). I, el número que usted debe digitar para seleccionar la categoría de mercancías deseada, se utiliza para seleccionar el elemento correcto de AO(), en donde se han de almacenar las perlas, figurillas y especias obtenidas. Asimismo, se emplea I como el segundo subíndice de EQ(,).

Observe que si en un posterior intercambio se obtiene una segunda cantidad de perlas, figurillas o especias, la cantidad se debe sumar a la cantidad ya almacenada en la matriz AO(). Tras registrar en AO() los bienes recientemente obtenidos, el bucle retorna, de ser necesario, al principio. Tras haber intercambiado todas las mercancías termina la actividad comercial y se le informa respecto a las cantidades de perlas, figurillas y especias que se han adquirido y cargado a bordo de su barco. Las líneas 10220 y 10244 imprimen la cantidad de cada artículo, almacenadas ahora en la matriz AO(). El control retorna entonces al programa principal.

En el próximo capítulo concluiremos el juego mercantil *Nuevo Mundo* con la adición de las secciones finales, en las que usted se verá implicado en una insurrección local antes de que emprenda la dilatada travesía de regreso al Viejo Continente y venda las mercancías recientemente adquiridas, esperando obtener suficientes beneficios como para justificar tan largo viaje y los avatares pasados durante la expedición.

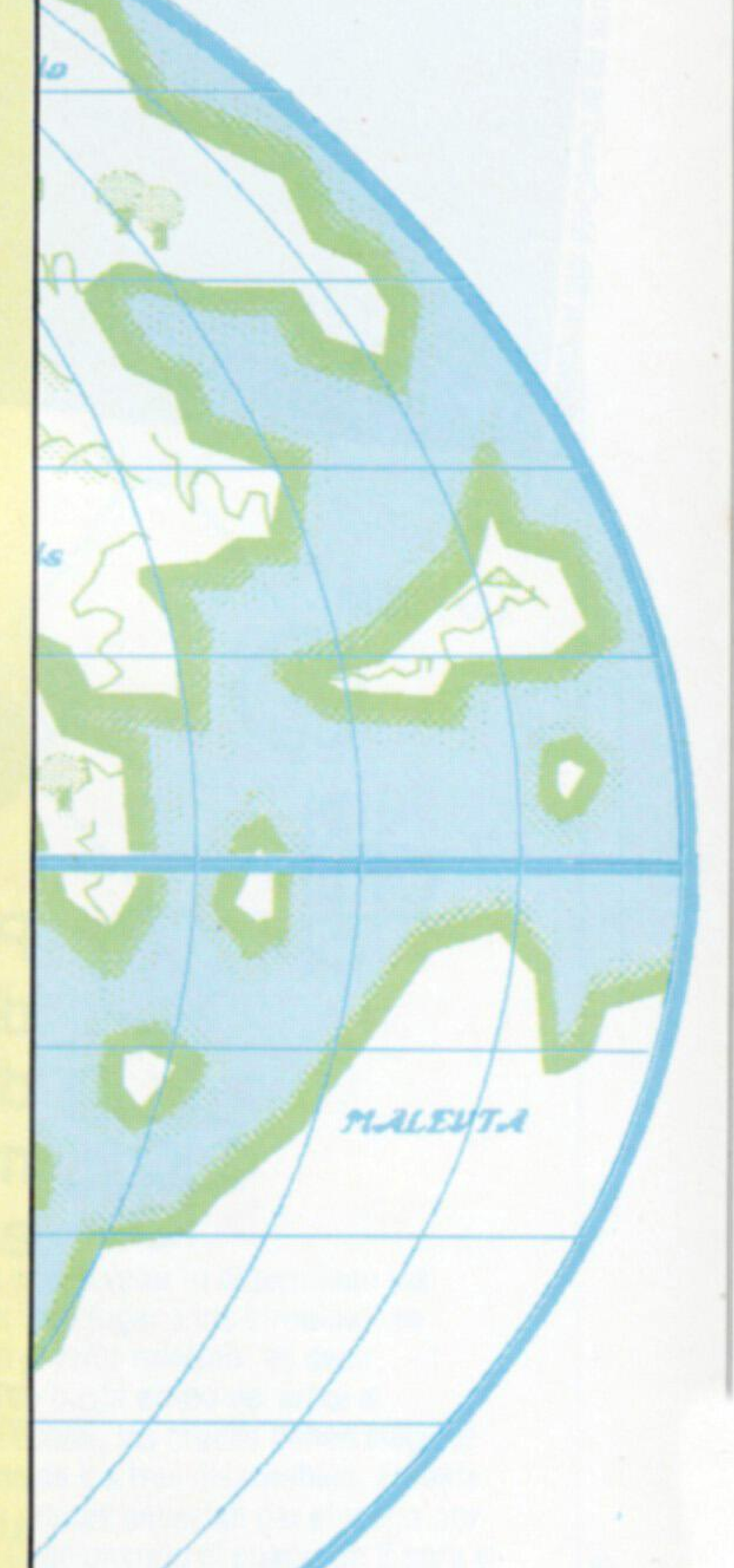
Módulo 12: Intercambio

Adición al cuerpo principal del programa

```
892 GOSUB 10070
```

Rutina de intercambio

```
10070 PRINT CHR$(147):GOSUB 9200:REM INTERCAMBIO
10072 IF OA(2)=0THEN 10080
10074 S$="EL JEFE NO QUIERE TUS ARMAS*":GOSUB 9100
10076 S$="PORQUE PODRIAN ACARREARLE PROBLEMAS*":GOSUB
      9100
10078 PRINT:GOSUB 9200
10080 IF OA(3)<>0 OR OA(4)<>0 OR OA(5)<>0 OR OA(6)<>0 THEN
      10100
10085 S$="NO TE QUEDA NINGUNA MERCANCIA*":GOSUB 9100
10090 S$="CON LA QUE COMERCIAR*":GOSUB 9100
10095 GOTO 10038
10100 S$="EN TRUEQUE POR LOS CUCHILLOS*":GOSUB 9100
10102 S$="SAL TELA O JOYAS QUE POSEAS*":GOSUB 9100
10104 S$="TE OFRECE PERLAS, FIGURILLAS*":GOSUB 9100
10106 S$="Y ESPECIAS*":GOSUB 9100
10108 PRINT:GOSUB 9200
10110 S$="CUANDO SALISTE DE PUERTO ESTAS*":GOSUB 9100
10112 S$="VALIAN:*":GOSUB 9100
10114 S$="PERLAS-2 P DE ORO CADA UNA*":GOSUB 9100
10116 S$="FIGURILLAS-2 P DE ORO CADA UNA*":GOSUB 9100
10118 S$="ESPECIAS-1 P DE ORO EL GRAMO*":GOSUB 9100
10120 PRINT:GOSUB 9200
10122 S$="PERO CUANDO VUELVAS A CASA*":GOSUB 9100
10124 S$="QUIZA ESTOS VALORES HAYAN CAMBIADO*";GOSUB
      9100
10125 PRINT:GOSUB 9200:S$=K$:GOSUB 9100
10126 GET I$:IF I$="" THEN 10126
10130 FOR T=3 TO 6
10135 IF OA(T)=0 THEN 10200
10140 PRINT CHR$(147):GOSUB 9200
10145 PRINT"TIENES";OA(T);
10150 IF T=3 THEN S$="SACOS DE SAL*"
10151 IF T=4 THEN S$="BALAS DE TELA*"
10152 IF T=5 THEN S$="CUCHILLOS*"
10153 IF T=6 THEN S$="JOYAS*"
10155 GOSUB 9100
10156 PRINT:GOSUB 9200
10160 S$="A CAMBIO EL JEFE TE OFRECE*":GOSUB 9100
10165 PRINT "YA SEA";OA(T)*EQ(T-2,1);"PERLAS"
10166 PRINT "O BIEN";OA(T)*EQ(T-2,2);"FIGURILLAS"
10167 PRINT""        0";OA(T)*EQ(T-2,3);"GRAMOS DE ESPECIAS"
10168 PRINT:GOSUB 9200
10170 S$="QUIERES PERLAS, FIGURILLAS*":GOSUB 9100
10172 S$="O ESPECIAS?*":GOSUB 9100
10174 S$="(ENTRA 1,2 o 3)*":GOSUB 9100
10175 INPUT I$
10176 I=VAL(I$):IF I<1 OR I>3 THEN 10174
10180 AO(I)=AO(I)+(OA(T)*EQ(T-2,I))
10190 PRINT:PRINT"LAS  ";T$(I);"  SE CARGAN EN EL BARCO"
10192 S$=K$:GOSUB 9100
10194 GET I$:IF I$="" THEN 10194
10200 NEXT
10210 PRINT:PRINT:GOSUB 9200
10215 S$="          FIN DEL INTERCAMBIO*":GOSUB 9100
10216 PRINT:GOSUB 9200
10218 S$="HAS OBTENIDO:*":GOSUB 9100
10220 PRINT AO(1);"PERLAS"
10222 PRINT AO(2);"FIGURILLAS"
10224 PRINT AO(3);"GRAMOS DE ESPECIAS"
10226 PRINT:GOSUB 9200
10228 S$=K$:GOSUB 9100
10229 GET I$:IF I$="" THEN 10229
10230 RETURN
```



Complementos al BASIC

Spectrum:

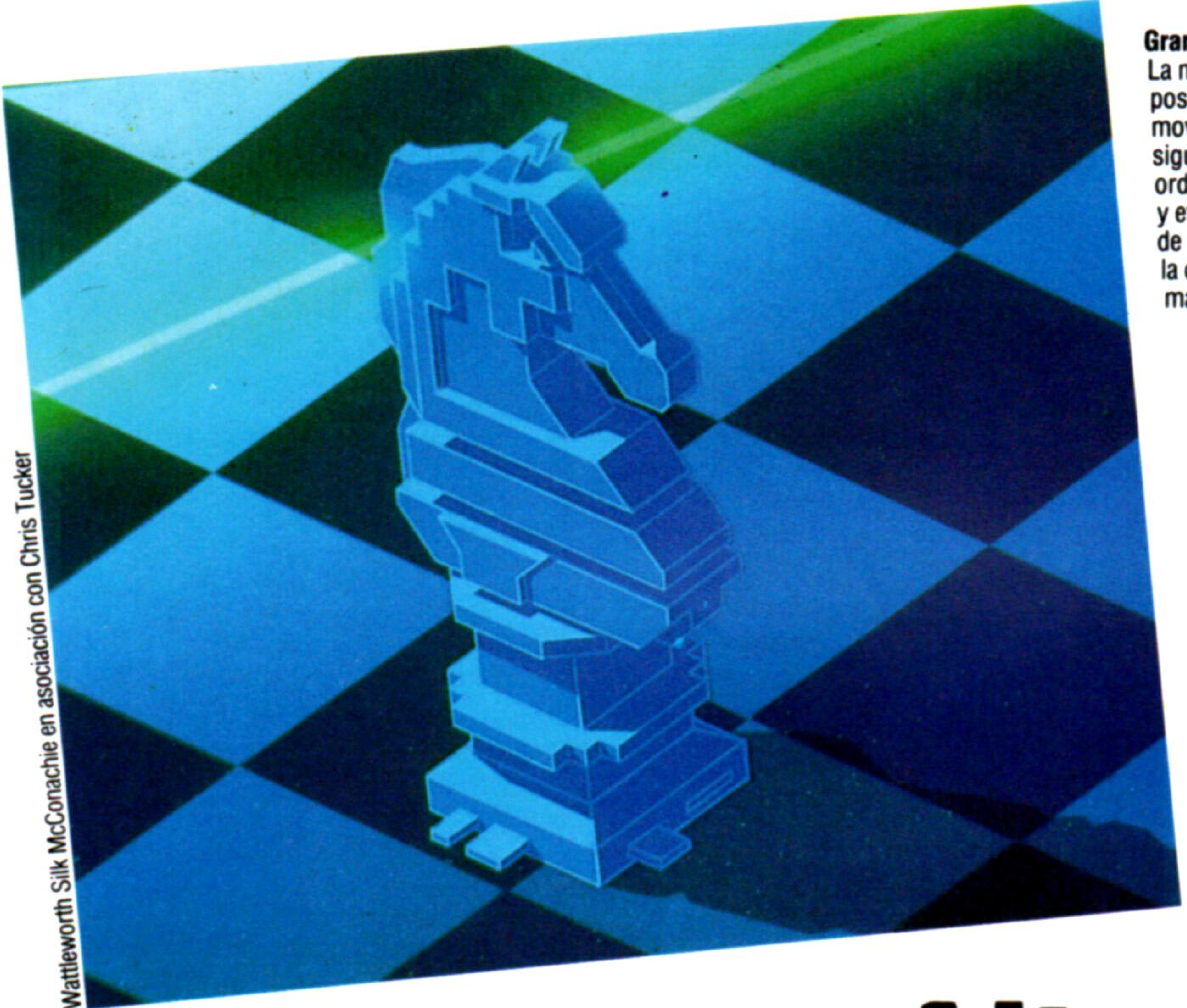
Reemplazar EQ(,) por Q(,) y AO() por E() en todo el listado e introducir estos cambios:

```
10070 CLS
10126 LET I$=INKEY$:IF I$="" THEN GO TO 10126
10140 CLS:GO SUB 9200
10229 LET I$=INKEY$:IF I$="" THEN GO TO 10299
```

BBC Micro:

Introducir las siguientes modificaciones:

```
10070 CLS
10126 I$=GET$
10140 CLS:GOSUB 9200
10229 I$=GET$
```



Wattleworth Silk McConachie en asociación con Chris Tucker

Gran maestro
La mayoría de los programas inteligentes de ajedrez poseen la capacidad de "anticipar" una cantidad de movimientos en el juego para evaluar cuál de los siguientes movimientos posibles es el mejor. Los ordenadores hacen esto construyendo un árbol de juego y efectuando una búsqueda en él. En los primeros días de la investigación en materia de AI, se consideraba que la capacidad de un ordenador para jugar al ajedrez era la máxima medida de la inteligencia de la máquina

Alta estrategia

Profundicemos en la mecánica de programas capaces de determinar los mejores movimientos en juegos de estrategia

La frase "juego por ordenador" suele traer a la mente una imagen de alienígenas a los que hay que capturar en la inmensidad del espacio o intentar atraer en cavernas subterráneas; pero no siempre es así. En los primeros días de la informática, algunos pioneros (entre ellos celebridades tales como Claude Shannon, John von Neumann y Alan Turing) dedicaron sus esfuerzos a programar un ordenador para que jugara al ajedrez.

El ajedrez se consideraba como el juego intelectual por excelencia, y un programa que jugara al ajedrez con éxito representaba la máxima prueba de la inteligencia de una máquina. En la actualidad hay sistemas de ordenador, como el *Belle* y el *Cray Blitz*, que juegan al nivel de los maestros internacionales, aunque pocas personas afirmarían que estas máquinas piensan. Aun así, el ajedrez y otros juegos de destreza mental ofrecen un campo ideal para poner a prueba teorías de planificación estratégica: la competición abierta.

La mayor parte de los programas de juegos de destreza se basan en técnicas de búsqueda arborescente, bastante similares a las descritas en el capítulo anterior, pero con modificaciones para tener en cuenta a un adversario. La idea fundamental es la de la "anticipación". El programa construye un árbol de juego considerando sus propios movimientos, analizando los contramovimientos que tiene a su disposición el oponente, anticipando sus respuestas a los mismos y así sucesivamente.

El diagrama del árbol de juego muestra el árbol anticipado para un juego imaginario entre dos personas. La raíz del árbol es la posición actual, con MAX listo para mover. Los nudos terminales, u hojas, son posiciones fin-del-juego. El árbol se utiliza para seleccionar el movimiento a efectuar mediante un procedimiento denominado *minimaximización*, que fue enunciado claramente por primera vez en 1949 por Claude Shannon. Trabaja asignando primero valores numéricos a los nudos terminales: supongamos uno para ganado, cero para empate y uno negativo para perdido. Estos valores se combinan luego al ir recorriendo el árbol, sobre el supuesto de que el jugador (MAX) siempre recoge los mayores mientras que el oponente (MIN) siempre elige los menores, para producir valores para nudos más altos.

En este ejemplo el valor de la raíz es 0, indicando que el juego producirá un empate (siempre y cuando ninguna de las partes cometa ningún error). El movimiento correcto en el nivel superior es, por tanto, M1, M3 o M4, pero no M2. Las reglas que rigen la bifurcación y generación de valores de nudos están determinadas por las reglas del juego específico. Sólo en los juegos triviales, como en el tres en raya, se puede construir todo el árbol del juego completo hasta el final. El ajedrez, por ejemplo, posee un *factor de ramificación* de alrededor de 32, lo que significa que en cualquier posición hay aproximadamente 32 movimientos legales. La anticipación de cuatro niveles (dos movimientos por cada lado) conduciría a más de un millón de nudos terminales. Esta "explosión combinatoria" significa que los programas para jugar al ajedrez no pueden anticiparse hasta la conclusión del juego.

En cambio, la mayoría de los programas de juegos se anticipan hasta donde pueden y evalúan las posiciones allí halladas. Para hacerlo se requiere un método para juzgar cuán favorables o desfavorables son los nudos de las hojas, aun cuando no se sepa el verdadero resultado. Ésta se suele denominar *función de evaluación estadística* e introduce necesariamente imprecisión, porque sólo es una estimación del resultado final. No obstante, la razón de ser de analizar una cierta distancia de anticipación y utilizar una función de evaluación imperfecta, es que se aplicará cerca del final del juego y será, con toda probabilidad, una estimación más acertada que la efectuada sin realizar búsqueda alguna.

Tomando a modo de ejemplo el juego de las damas, podríamos elaborar una función de evaluación muy simple de cuatro términos basada en:

D Ventaja de dama
P Ventaja de pieza

M Diferencia de movilidad
C Control del centro

Estos atributos se pueden calcular examinando el tablero. Por ejemplo, D=WD−BD, donde WD es la cantidad de damas defensoras y BD representa la cantidad de damas oponentes.

Las otras variables reflejan varios hechos: es mejor tener más piezas que el oponente (el perdedor acaba sin ninguna pieza); es útil disponer de mayor movilidad, y los cuadrados centrales, en las damas así como en el ajedrez, son más valiosos que los cuadrados laterales. El programa debe combinar de alguna forma estas cantidades en un marcador global.

Suponiendo que decidimos que una dama (D) es tres veces más valiosa que una pieza común (P), que una pieza extra vale dos movimientos y medio adicionales (M) y que un movimiento de más es el doble de meritorio que controlar un cuadrado central más, nuestra función de evaluación sería:

V=15D+5P+2M+C

(Típicamente se utilizan estimaciones de enteros por la mayor velocidad de cálculo.)

Ésta, sin embargo, es una función de evaluación muy torpe. A modo de comparación, el clásico programa de Arthur Samuel para jugar a las damas, de principios de los años sesenta, empleaba hasta 25 parámetros. Aquí los coeficientes son, asimismo, bastante arbitrarios. Parte de la diversión al desarrollar programas de juegos reside en ajustar tales estimaciones para obtener un buen equilibrio. Uno de los puntos más exquisitos del programa de Samuel era que regulaba sus propias estimaciones automáticamente, lo que constituía una clase rudimentaria de aprendizaje.

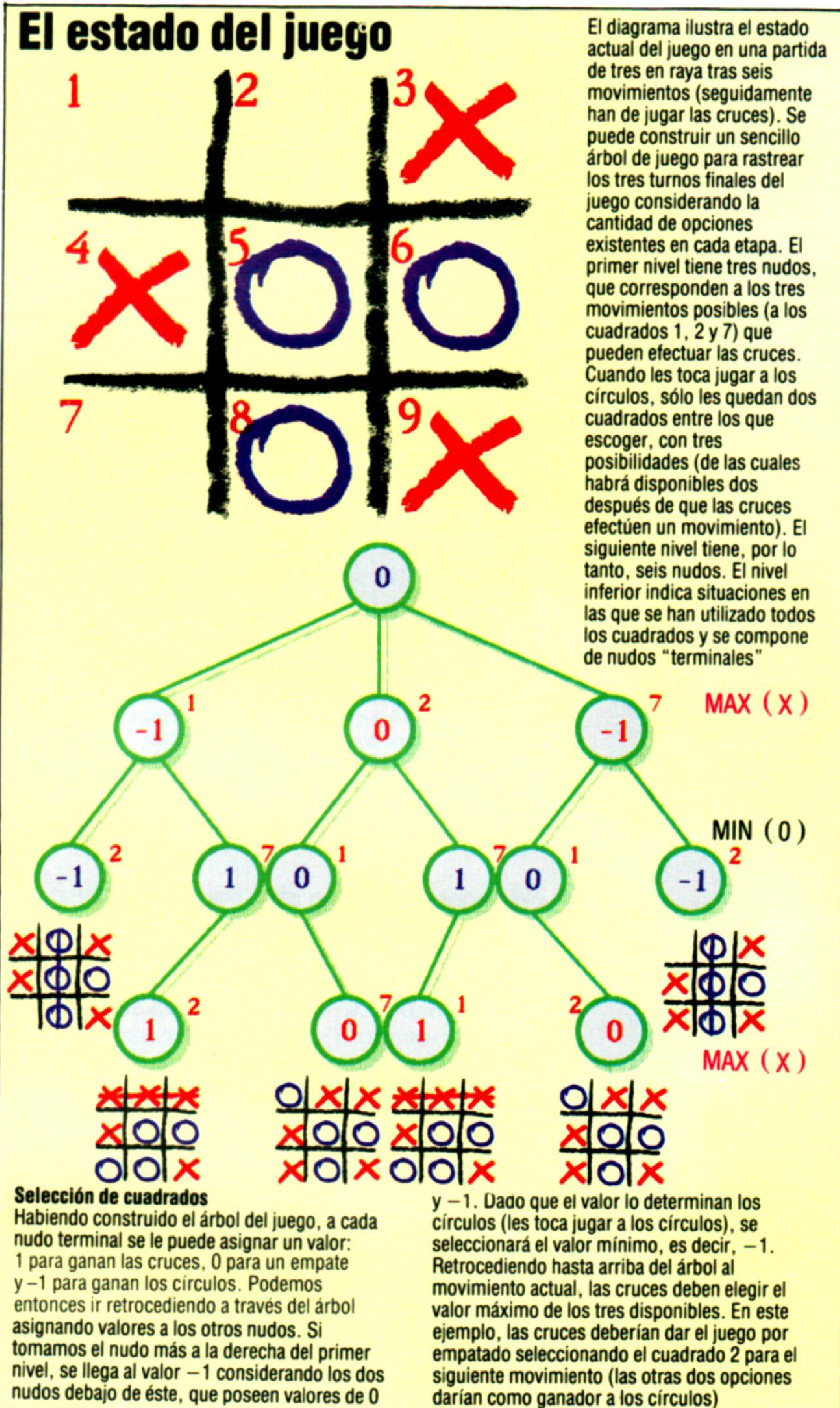
En el transcurso de los últimos treinta años, la idea de otorgar valores numéricos a las características del juego y combinarlos en una suma estimada para calcular el valor de cualquier posición ha demostrado su valor. La función de evaluación juega un papel similar al de la medición de distancia heurística en la solución de problemas mediante la búsqueda, analizada anteriormente.

Un programa que simplemente anticipe una profundidad fijada y evalúe los nudos terminales que se han hallado, se encontrará con problemas. Esto se debe a que algunas posiciones de los juegos son "tranquilas" mientras que otras son sumamente "inestables". En ajedrez es probable que tras una captura o coronación de peón el estado del juego sea muy inestable: en el siguiente movimiento se podría producir una recuperación. Si esto tiene lugar un nivel más allá del "horizonte" del programa, la evaluación será sumamente engañosa.

Para reducir este efecto, no poseen una anticipación de profundidad fija. Poseen una medida de "inmovilidad", que indica si una posición se puede evaluar de forma fiable. Si la evaluación no es segura, la búsqueda se empuja un poco más hacia adelante. En ajedrez y damas, esto implica examinar secuencias de capturas comparativamente largas.

El algoritmo *alfa-beta* apareció por primera vez en 1967, en el programa MacHack de Greenblatt. Es un refinamiento de la minimaximización básica y ofrece el mismo resultado pero con muchísimo menos esfuerzo. El diagrama de la página siguiente

El estado del juego



El diagrama ilustra el estado actual del juego en una partida de tres en raya tras seis movimientos (seguidamente han de jugar las cruces). Se puede construir un sencillo árbol de juego para rastrear los tres turnos finales del juego considerando la cantidad de opciones existentes en cada etapa. El primer nivel tiene tres nudos, que corresponden a los tres movimientos posibles (a los cuadrados 1, 2 y 7) que pueden efectuar las cruces. Cuando les toca jugar a los círculos, sólo les quedan dos cuadrados entre los que escoger, con tres posibilidades (de las cuales habrá disponibles dos después de que las cruces efectúen un movimiento). El siguiente nivel tiene, por lo tanto, seis nudos. El nivel inferior indica situaciones en las que se han utilizado todos los cuadrados y se compone de nudos "terminales"

Selección de cuadrados
Habiendo construido el árbol del juego, a cada nudo terminal se le puede asignar un valor: 1 para ganan las cruces, 0 para un empate y −1 para ganan los círculos. Podemos entonces ir retrocediendo a través del árbol asignando valores a los otros nudos. Si tomamos el nudo más a la derecha del primer nivel, se llega al valor −1 considerando los dos nudos debajo de éste, que poseen valores de 0 y −1. Dado que el valor lo determinan los círculos (les toca jugar a los círculos), se seleccionará el valor mínimo, es decir, −1. Retrocediendo hasta arriba del árbol al movimiento actual, las cruces deben elegir el valor máximo de los tres disponibles. En este ejemplo, las cruces deberían dar el juego por empatado seleccionando el cuadrado 2 para el siguiente movimiento (las otras dos opciones darían como ganador a los círculos)

nuestra parte de un árbol de juego entre dos jugadores llamados MINI y MAX.

Las letras junto a cada nudo (de la A a la L) muestran el orden por el cual se examina el árbol, utilizando un procedimiento de primero en profundidad; los números son evaluaciones. Las barras simples señalan lo que se conoce como *limitaciones alfa* y las barras dobles denotan *limitaciones beta*. Éstas cercenan las bifurcaciones que no pueden incidir en el resultado final.

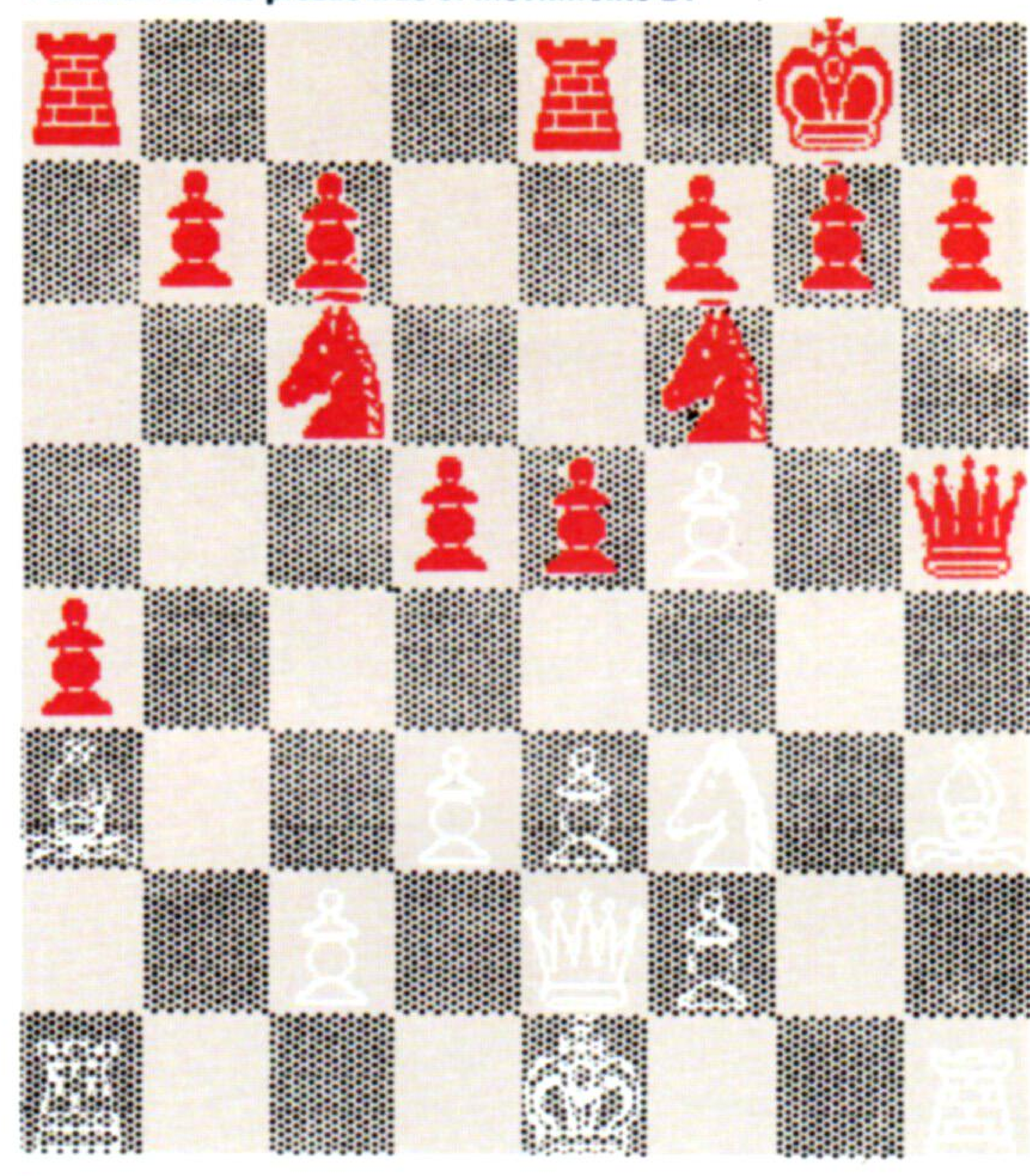
Una limitación *alfa* se produce en el nudo E, que no es necesario evaluar nunca, como tampoco a ninguno de sus descendientes, si hubiera alguno. Para cuando llegamos al nudo E sabemos que el nudo C obtiene un marcador de 15, pero en el nudo D el oponente puede forzarnos a bajar a 10. No tiene ningún sentido averiguar si nos podemos ver

Movimientos de la partida

1	d2-d3	e7-e5
2	Cb1-d2	Cb8-c6
3	g2-g3	Cg8-f6
4	Af1-g2	Af8-c5
5	e2-e3	d7-d5
6	Cg1-e2	0-0
7	a2-a3	Ac8-f5
8	b2-b3	Cf6-g4?!
9	h2-h3	ng4-f6
10	Ac1-b2	Dd8-d6
11	g3-g4	Af5-e6
12	Ce2-g3	a7-a5
13	Dd1-e2	Cf6-d7
14	Cg3-f5	Ae6xf5
15	g4xf5	Cd7-f6
16	h3-h4	Tf8-e8
17	Ag2-h3	a5-a4
18	b3-b4	Ac5xb4!?
19	a3xb4	Dd6xb4
20	Ab2-a3	Db4xh4
21	Cd2-f3	Dh4-h5
22	Cf3-d2	Dh5xe2+?
23	Re1xe2	b7-b5
24	c2-c3	h7-h6
25	Ah3-g2	Ta8-a5
26	Ta1-b1	Te8-b8
27	Tb1-b2	Tb8-b6
28	Th1-b1	Rg8-h7
29	Aa3-c5	Tb6-b8
30	Ac5-a3	Cc6-a7
31	Cd2-f3	Cf6-d7
32	Cf3-e1	c7-c6
33	Cd1-c2	Tb8-a8
34	Cc2-b4	Ta8-d8
35	Cb4-a2	Ca7-c8
36	c3-c4!	d5xc4
37	d3xc4	b5xc4
38	Ag2xc6	Cc8-a7
39	Ac6-e4	Cd7-f6
40	Aa3-e7	Td8-c8
41	Ae7xf6	g7xf6
42	Tb1-b6!	c4-c3
43	Tb6xf6	Rh7-g7
44	Tf6-b6	a4-a3
45	Tb1-g1+	

En este punto el programa indicó que estaría 2.4 peones abajo y los programadores abandonaron

Posición de las piezas tras el movimiento 21



Esta partida forma parte de una serie en la que el programa de ajedrez por ordenador más potente del mundo, el *Cray Blitz*, fue derrotado 4-0 por David Levy, de Intelligent Software, tras una apuesta de 5 000 dólares entre el señor Levy y los programadores del *Blitz*. Aunque el programa ha obtenido la categoría ajedrecista de National Master, el juego demuestra que aún es necesario muchísimo trabajo para que los ordenadores supongan un serio desafío a los mejores jugadores humanos del mundo

obligados a bajar aún más, puesto que obviamente esta ruta es menos deseable que la ruta a través del nudo C. Por tanto, se pueden sustraer de toda consideración los otros descendientes del nudo F.

En el nudo I se aplica el mismo razonamiento pero al contrario. Para cuando llegamos allí, sabemos que G produce un marcador de 20. El nudo H, con 25, parece mejor, pero MINI (y no MAX) elige entre los nudos G y J, y claramente se inclinará por G. Por consiguiente, no hay necesidad de ver si I es aún más prometedor, puesto que a MAX jamás se le permitirá llegar allí.

Podemos expresar estas ideas desde el punto de vista de un árbol genealógico. MAX es un machista que piensa que el nudo C, por ejemplo, es el tío de los nudos D y E, que son ambos hijos del mismo padre. MINI, por su parte, es una feminista y, en cuanto a ella respecta, G es la tía de las hermanas H e I, cuya madre es J. En la medida en que usted no se sienta confundido por nudos que cambian de sexo en niveles alternativos, esta analogía nos permitirá explicar con precisión la regla *alfabeta*:

- Apenas descubre MAX un hijo que es *peor* que cualquiera de sus tíos, ignora a los otros hermanos de ese hijo.
- Apenas descubre MINI una hija que es *mejor* que cualquiera de sus tías, ignora a las otras hermanas de esa hija.

En el mejor de los casos, el algoritmo *alfa-beta* sólo examina dos veces más la raíz cuadrada de la cantidad de nudos terminales del árbol del juego, en comparación con la simple minimaximización. En el peor de los casos, examina la misma cantidad, y de forma ligeramente más lenta. Para evitar el primero de nuestros dos casos, es importante generar los hermanos y hermanas de cada nivel por un orden sensato. En los niveles maximizadores, se los debe generar por orden de primero el mejor, y en los niveles minimizadores, por orden de primero el peor (primero el mejor para el adversario).

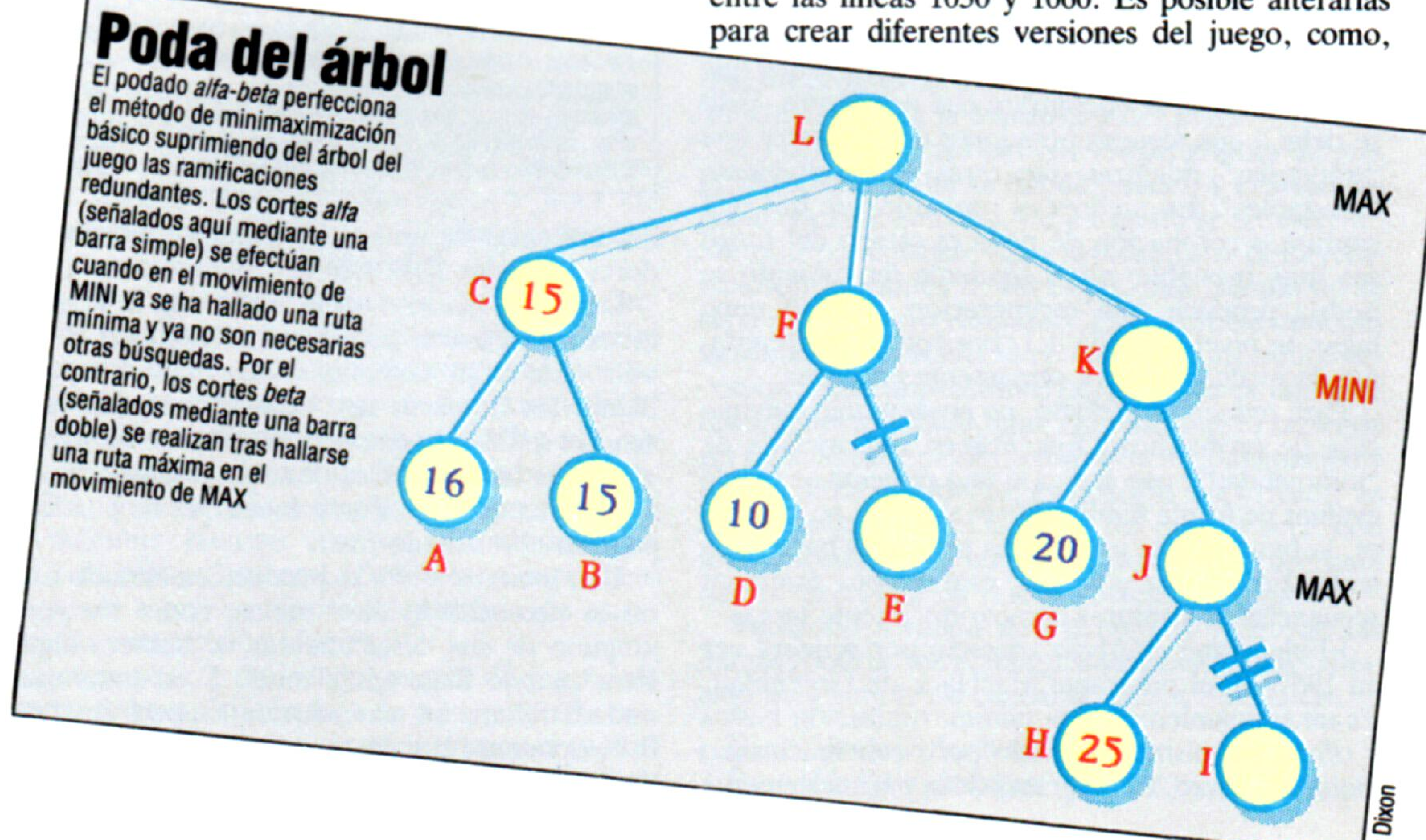
Para ilustrar los importantes conceptos de la búsqueda arborescente, presentamos un juego artificial que comprende técnicas de búsqueda casi puras. Ello significa que los detalles de representación del tablero, generación de movimientos y evaluación estática (que son cruciales para el éxito de cualquier programa verdadero pero que también son específicos de cada juego) no oscurecen la simplicidad esencial del procedimiento *alfa-beta*.

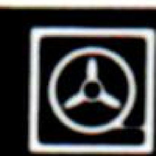
No hay ni tablero ni ninguna pieza: el estado del juego se describe completamente mediante un único número, retenido como V%. Para el ordenador, el objetivo del juego es que V% alcance el valor 255, y para usted conseguir que sea menor que −255.

A cada turno el jugador puede elegir entre aplicar una de cuatro funciones (A,B,C,D), listadas entre las líneas 1030 y 1060. Es posible alterarlas para crear diferentes versiones del juego, como,

Poda del árbol

El podado *alfa-beta* perfecciona el método de minimaximización básico suprimiendo del árbol del juego las ramificaciones redundantes. Los cortes *alfa* (señalados aquí mediante una barra simple) se efectúan cuando en el movimiento de MINI ya se ha hallado una ruta mínima y ya no son necesarias otras búsquedas. Por el contrario, los cortes *beta* (señalados mediante una barra doble) se realizan tras hallarse una ruta máxima en el movimiento de MAX





por ejemplo, hacer que le resulte más difícil al ordenador.

El juego es trivial pero ilustra la estrategia de búsqueda sin que detalles ajenos dificulten su apreciación. Además, es sumamente matemático, de modo que el ordenador posee una ventaja incorporada. En esta versión, la minimaximización *alfa-beta* se basa en gran medida en el uso de funciones recursivas con parámetros y variables locales. Los parámetros son los siguientes:

VV% Estado actual del juego
A% Mejor *Alfa* hasta ahora a este nivel
B% Peor *Beta* hasta ahora a este nivel
D% Indicador de profundidad

Dado que el programa implica recursión, aquí sólo incluimos una versión en BASIC BBC.

El juego de los números

```
10 REM ****************************************
11 REM **              Listado 3.1:          **
12 REM **        EL JUEGO DE LOS NUMEROS     **
13 REM ****************************************
50 MODE 7
100 REM — Juego para demostrar busqueda:
120 GOSUB 1000   :REM inicializacion
130 GOSUB 1600   :REM instrucciones
150 REPEAT
160    GOSUB 2000   :REM preparar nuevo juego
170    INPUT "Quien va primero (1=Tu, 2=Yo)  ",H1%
180    IF H1%<1 OR H1%>2 THEN GOTO 170
200    REM — bucle principal:
210    REPEAT
220       IF H1%=1 THEN GOSUB 3000
230       REM — turno de la persona.
240       GOSUB 3500   :REM visualizacion tablero
250       H1%=1 : REM siempre 1 tras 1.er ciclo
260       GOSUB 4000 : REM verificar ganador
270       IF EG%=0 THEN GOSUB 5000
280       REM — turno del ordenador.
290       GOSUB 3500 : REM mostrar estado del juego
300       GOSUB 4000 : REM verificar final del juego
310       UNTIL EG%<>0 OR M%>33
320    REM — ultimo movimiento:
330    GOSUB 6000 : REM felicitaciones
340    PRINT"Otra partida (N=No)  ";
350    Y$=GET$
360    UNTIL Y$="N" OR Y$="n"
365 PRINT
370 PRINT"Hasta otra, y gracias por jugar!"
400 END
444 :
500 DEF FNmaximov(VV%,A%,B%,D%)
505 REM --------- mi turno:
510 LOCAL P%,E%, KEEP%
515 IF D%>=MD% OR ABS(VV%)>HI% THEN=VV% : REM valor estatico
520 REM si no profundizar mas:
525 P%=0
530 REPEAT P%=P%+1
535    H%=P%: V%=VV%: GOSUB 5500:REM efectuar movimiento
536    IF D%=1 THEN PRINT CHR$(H%+64);" = ";
540    E%=FNminimov(V%,A%,B%,D%+1)
545    IF E%>A% THEN A%=E%: KEEP%=P%
548    IF D%=1 THEN PRINT E%;";  ";
550    UNTIL P%>3 OR A%>=B%
555 IF A%>BV% AND D%=1 THEN BV%=A%: HH%=KEEP%
556 REM guardar el mejor hasta ahora.
560 =A%
565 REM volver con max A%
570 :
700 DEF FNminimov(VV%,A%,B%,D%)
710 REM --------- turno del otro:
720 LOCAL E%,P%
730 IF D%>=MD%OR ABS(VV%)>HI% THEN=VV%
740 P%=0
750 REPEAT P%=P%+1: H%=P%
755    V%=VV%: GOSUB 5500: REM efectuar movimiento
760    E%=FNmaximov(V%,A%,B%,D%+1)
770    IF E%<B% THEN B%=E%
780    UNTIL P%>3 OR B%<=A%
790 =B%
796 REM vuelve con el valor de B% mas bajo
999 :
1000 REM -- rutina inicializadora:
1001 BL$=""
1002 @ %=4
1020 REM -- las 4 funciones:
1030 DEF FNA(X%)=2*X%-7
1040 DEF FNB(X%)=X% DIV 2+1
1050 DEF FNC(X%)=-4*X%+17
1060 DEF FND(X%)=3*X%-4
1070 LO%=-255: HI%=255
1150 RETURN
1160
1600 REM -- rutina instrucciones:
1610 CLS:PRINT
1620 PRINT""Bienvenido al Juego de los Numeros!"
1630 PRINT"Si no conoces las reglas,"
1635 PRINT"LEETE EL CAPITULO!"'
1636 PRINT"Nota: Yo maximizo : tu minimizas."
1640 PRINT"Para ver el efecto de un movimiento, digita:"
1645 PRINT"A, B, C, o D. Para efectuarlo, digita X."
1650 PRINT:PRINT"Buena suerte!";CHR$(7)
1660 RETURN
1670
2000 REM -- Rutina de preparacion:
2010 M%=0 : REM movimientos
2020 V%=RND(15)-8: REM estado inicial.
2050 EG%=0
2060 PRINT'"Estado inicial=";V%
2100 RETURN
2110
3000 REM -- Movimiento de la persona
3010 M%=M%+1
3020 PRINT
3030 REPEAT
3040    PRINT"Que movimiento haces ? ";
3050    H$=GET$: PRINT H$;
3060    IF H$="A" THEN PRINT FNA(V%): H%=1
3070    IF H$="B" THEN PRINT FNB(V%): H$=2
3080    IF H$="C" THEN PRINT FNC(V%): H$=3
3090    IF H$="D" THEN PRINT FND(V%): H$=4
3100    UNTIL H$="X"
3120 GOSUB 5500 : REM elegir H%
3150 RETURN
3160 :
3500 REM -- rutina visualizacion del tablero:
3520 CLS: PRINT
3522 PRINT"      Movimiento  ";M%;" -->  ";
3523 IF M%<1 THEN RETURN
3525 PRINT CHR$(64+H%);
3530 PRINT "  =  ";V%
3535 RETURN
3700
4000 REM -- Rutina comprobacion ganador (sobre M$):
4001 IF M%<1 THEN RETURN
4010 EG%=0
4020 IF V%<LO% THEN EG%=-1
4030 IF V%>HI% THEN EG%=1
4040 RETURN
4050 :
5000 REM -- Rutina movimiento del ordenador:
5005 W%=V% : REM guardar estado actual.
5010 M%=M%+1
5015 MD%=6 : REM max profundidad
5020 IF M%<4 THEN MD%=4
5030 IF M%>8 THEN MD%=8
5040 GOSUB 5200 : REM -->H%
5045 V%=W% : REM restablecer estado.
5050 GOSUB 5500 : REM hacerlo.
5070 RETURN
5080 :
5200 REM -- Seleccion de movimiento:
5210 BV%=LO% : D%=0
5220 BV%=FNmaximov(V%,LO%,HI%,1)
5230 H%=HH%
5240 PRINT "Pulsar cualquier tecla para continuar: ";
5244 C%=GET
5250 RETURN
5270 :
5500 REM -- Rutina efectuar movimiento (H% : V%):
5505 ON H% GOTO 5510,5520,5530,5540
5510 V%=FNA(V%): RETURN
5520 V%=FNB(V%): RETURN
5530 V%=FNC(V%): RETURN
5540 V%=FND(V%): RETURN
5550 :
6000 REM -- Rutina de felicitaciones:
6010 PRINT "EL JUEGO HA TERMINADO!"
6020 IF EG%>0 THEN PRINT "He ganado yo!!"
6030 IF EG%<0 THEN PRINT "Bien hecho!"
6040 IF EG%=0 THEN PRINT "Ha sido empate"
6050 RETURN
6600 :
```

Prólogo

Parece que haya pasado mucho tiempo desde que los japoneses afirmaron que cambiarían el curso de la tecnología del ordenador invirtiendo dinero y trabajo de investigación en su proyecto de ordenadores de la quinta generación. Quizás una de las decisiones más significativas que adoptaron fue utilizar el PROLOG, lenguaje de programación poco conocido, como el "lenguaje central" de las máquinas de bases de datos inteligentes y de elevado rendimiento que prevén.

PROLOG representa las siglas de "*pro*gramming in *log*ic" (programar en lógica) y constituye una concreción buena pero imperfecta de ese ideal. Pero ¿por qué habría de ser deseable programar en lógica? Existen muchas lógicas que se pueden aplicar para describir el mundo y aspectos del mismo. Con algunas de éstas usted ya estará familiarizado, como la matemática, mientras que otras pueden parecerle bastante ajenas, como ciertas doctrinas filosóficas. El cálculo de predicado de primer orden es una lógica bastante parecida a la que utilizamos para el pensamiento y el análisis cotidiano, aunque posee su propia notación y ciertas restricciones. Se puede concebir el predicado como una relación entre cosas. En la frase "a Juan le gusta Ana", el predicado es "gusta" Podríamos escribir esto como "gusta (Juan, Ana)", que es menos inteligible pero más claro en cuanto a qué es el predicado y cuáles son sus argumentos. Para expresar que Juan es varón, podríamos escribir "varón(Juan)", donde "varón" es un predicado que toma un argumento ("Juan"). De modo similar, "mujer(Ana)", significa que Ana es una mujer.

Lo que tenemos aquí son simplemente declaraciones de hechos pero podemos ampliar la lógica para mostrar cómo algunos hechos *implican* otros. Utilizando la lógica de predicado, podemos describir el mundo en términos de hechos e implicaciones y usar nuestra descripción para deducir hechos nuevos a partir de los antiguos. Ello lo hacemos mediante la introducción de variables. Las variables lógicas son muy similares a aquellas que usted ya conoce a través del BASIC o algunos otros lenguajes de programación, con la excepción de que su esfera de acción se limita a la cláusula (hecho o implicación) en la que aparecen, en vez de a todo el conjunto de cláusulas. Esto significa que el Juan a quien le gusta Ana tal vez sea un Juan diferente del Juan que es un varón.

Ahora podríamos escribir un hecho tal como "mujer(X) →gusta(Juan, X)". La flecha significa "implica que", de modo que podemos leer esto como una regla que afirma que "el hecho de que X sea mujer implica que a Juan le gusta X". En castellano corriente, la regla se lee como "a Juan le gusta X si X es mujer". Esta regla es un ejemplo de un tipo especial de cláusula del cálculo de predicado que se denomina *cláusula horn* (trompeta). Las cláusulas *horn* poseen una sentencia como encabezamiento (el *consecuente*), que es verdadera sólo si todas las sentencias del cuerpo (los *antecedentes*) lo son.

A si B y C y D

es una cláusula *horn* con la cabeza A y el cuerpo B, C y D. Un hecho simple se puede considerar como un consecuente sin antecedentes y se lo presume verdadero.

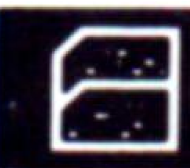
Con lógica podemos expresar el programa que deseamos escribir como un conjunto de hechos y reglas que describen las cosas en las que estamos interesados. Esta descripción es una forma bastante similar a aquella en que concebimos realmente el problema. Para "ejecutar" nuestro programa lógico intentamos demostrar la veracidad o falsedad de algún enunciado. Si éste es un hecho simple, entonces podemos suponer que es verdadero sin ningún otro esfuerzo. Si es el consecuente de alguna regla, entonces debemos abocarnos a demostrar la veracidad de todos sus antecedentes antes de que podamos afirmar que es verdadero. Por tanto, si deseamos saber si Juan es varón, intentamos demostrar la sentencia "varón(Juan)", que sabemos que es verdadera porque es un hecho que ya poseemos. Si queremos saber si a Ana le gusta Juan, primero necesitamos probar las sentencias "varón(Juan)" y "gusta(Juan, Ana)".

El PROLOG lo desarrolló A. Colmerauer en la Universidad de Marsella a principios de los años setenta, basándose parcialmente en el trabajo de Bob Kowalski, que en la actualidad se desempeña en el Imperial College de Londres. Utiliza sólo las cláusulas *horn* de la lógica de predicado y una notación similar a la que hemos



Preludio de "Complementos al PROLOG"
Al objeto de implementarse con eficacia, el PROLOG tiende a exigir generosas cantidades de RAM, y son raras las versiones del lenguaje que puedan ejecutarse en un ordenador personal. Los usuarios del Spectrum, no obstante, pueden considerar la adquisición del MICRO-PROLOG, escrito por Logic Programming Associates y distribuido por Sinclair en cassette. Los usuarios de otros micros pueden mantener vivas sus esperanzas, a la vista de que el PROLOG está captando rápidamente la atención de los productores de software y ya existen planes para versiones del lenguaje destinadas al BBC Micro y al Enterprise. El MICRO-PROLOG Spectrum difiere en muchos sentidos del DEC-10 PROLOG estándar, la versión utilizada en los ejemplos a lo largo de nuestra serie. No obstante, iremos imprimiendo una serie de breves *Complementos al* PROLOG, para que los usuarios del Spectrum puedan entrar listados directamente con MICRO-PROLOG. Las diferencias más importantes las explicaremos más adelante

Liz Heaney



mostrado más arriba. Se podría haber implementado el cálculo de predicado en su totalidad, pero el PROLOG, como todos los lenguajes de programación, representa un equilibrio entre eficiencia de proceso y poder de expresión. Se está desarrollando un gran trabajo de investigación para refinar este equilibrio, pero en su forma actual el PROLOG parece bastante estable.

Lo más cercano a un PROLOG estandarizado es una versión denominada DEC-10 PROLOG (porque se implementó por primera vez en un ordenador central de Digital Equipment Corporation), y la biblia de los usuarios del PROLOG es un libro de Clocksin y Mellish que con toda sencillez se titula *Programming in* PROLOG y describe esta versión y ofrece, paralelamente, muchos detalles prácticos. Casi todos los PROLOG actuales, incluyendo el C-PROLOG, están modelados según este estándar, aunque abundan los complementos y los dialectos. Para el micro, hay dos versiones que marchan a la vanguardia: la de Expert Systems, que se acerca mucho al estándar, y el MICRO-PROLOG, de Logic Programming Associates, que difiere en cuanto a sintáxis y estructura interna.

Las implementaciones de PROLOG utilizan muchísima memoria y por este motivo no caben con comodidad en micros con menos de 64 K de RAM. Sin embargo, existe el MICRO-PROLOG para el Spectrum y continúan apareciendo nuevas implementaciones.

Los programas en PROLOG no exhiben el familiar flujo de control en el cual se ejecuta la primera sentencia del programa, luego la segunda, y así sucesivamente hasta la última, con ocasionales bifurcaciones y bucles a lo largo del camino. En cambio, el PROLOG utiliza una técnica *"de retroceso"*.

Para resolver un interrogante, el PROLOG va trabajando hacia abajo a través de una cadena de reglas, planteándose a sí mismo un nuevo objetivo a demostrar en cada ocasión. Si un camino determinado a través de la cadena demuestra ser improductivo, el PROLOG "retrocederá" hasta un punto de elección anterior y después se bifurcará en una nueva dirección.

Esta forma de proceder le confiere al PROLOG un "tacto" muy diferente en comparación con cualquier otro lenguaje de programación. Los defensores de la programación lógica resaltan la naturaleza declarativa de un programa en PROLOG. Es decir, la lectura de una regla como si fuera una cláusula del cálculo de predicado de primer orden; por ejemplo, X es tío de Y si X es varón y X es el hermano de Z y Z es el padre de Y. Y el PROLOG siempre se puede leer en el estilo procedural más familiar (para demostrar que X es un tío de Y, demostrar primero que X es varón, después demostrar que X es el hermano de Z, y después demostrar que Z es el padre de Y). La disponibilidad de la lectura declarativa es algo de lo que carecen la mayoría de los otros lenguajes y, ciertamente, es una característica muy valiosa para ayudarlo a usted a comprender y, por lo tanto, a diseñar y comprobar sus programas.

Árbol de conocimiento

Las primeras aplicaciones de inteligencia artificial por lo general estaban dirigidas a la solución de problemas abstractos en los campos de la matemática y la física. Debido a que las áreas de conocimiento involucradas se podían formular fácilmente en términos de leyes fundamentales, se podían escribir programas para demostrar o refutar ciertos teoremas para poder resolver los problemas. Sin embargo, programar un ordenador para tratar "inteligentemente" problemas tomados del mundo real, en donde las causas y los efectos implícitos no se pueden expresar fácilmente en términos axiomáticos, exige un enfoque diferente. Para funcionar en estas circunstancias, un programa requerirá no sólo una base de datos que contenga hechos relevantes, sino también algún medio de relacionar estos hechos entre sí. El método más común de representar hechos y sus relaciones en el interior de un ordenador es el que se reproduce en la *red semántica*, una representación del conocimiento que se puede apreciar mejor como una estructura arborescente simple, tal como vemos. La red semántica se compone de *nudos* y *enlaces*, de los cuales los más obvios son ES-UN o ES-UN-TIPO-DE. En el ordenador, una red como ésta se puede representar como una estructura de datos, tal vez una matriz en BASIC o una cláusula en PROLOG

árbol
ES-UN
ES-UN
de hoja perenne
de hoja caduca
ES
ES
pino escocés
acebo
roble
plátano

Ian McKinnell

Ajustando la mira

Ahora montaremos los componentes del brazo y agregaremos los potenciómetros y la mira del trazador

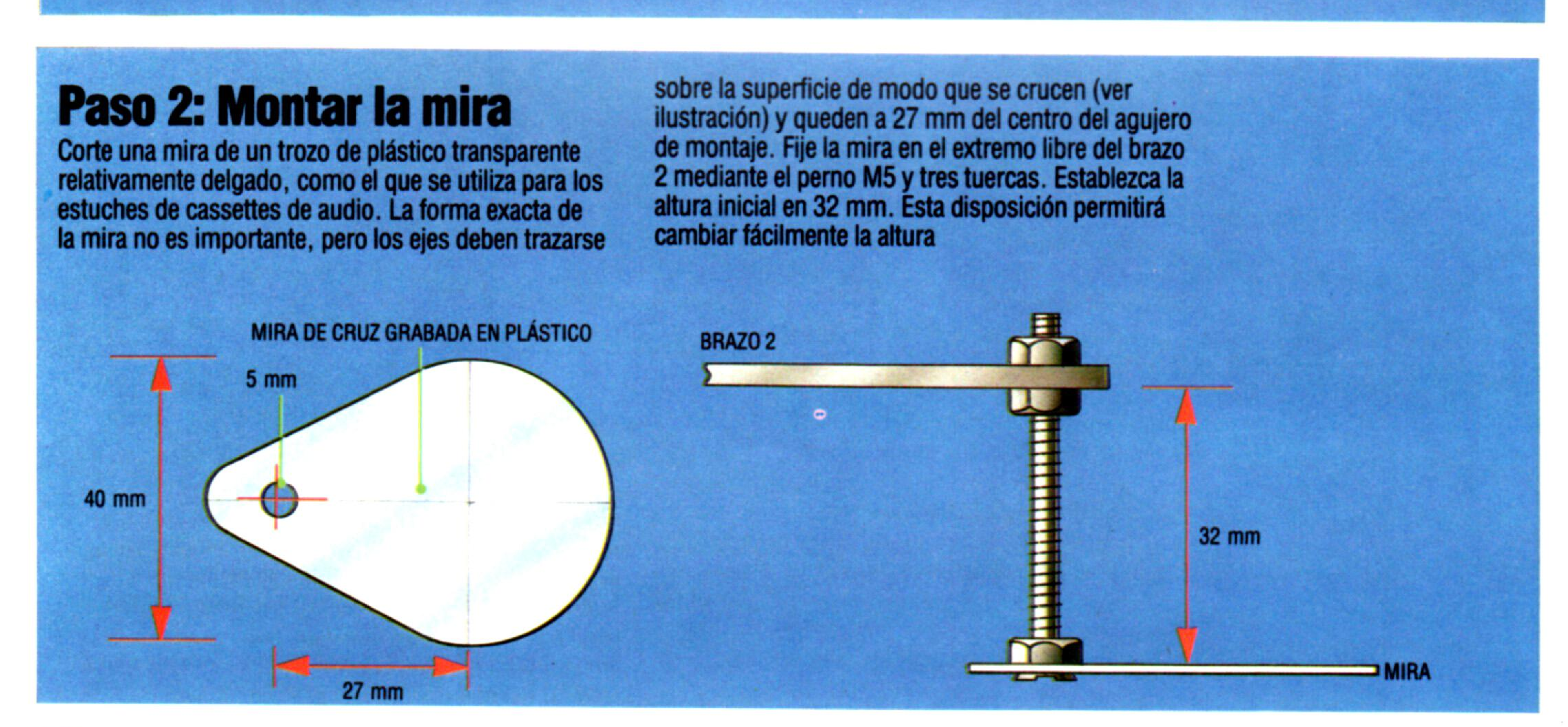
Paso 1: Ensamblar el brazo

Las dos piezas del brazo que cortamos en el último capítulo se engoznan mediante un par de potenciómetros, uno de los cuales se monta en la tapa de la caja del componente plástico y el otro a través de los extremos del brazo conector. Corte el husillo de cada potenciómetro de modo que sólo sobresalga 15 mm del cuerpo. Fije un potenciómetro en la tapa de la caja y otro en el brazo 2, ajustándolos en su sitio con las tuercas. Observe que los puntos de conexión del potenciómetro de la caja deben apuntar hacia el lado de la caja más cercano, mientras que los del otro deben apuntar a lo largo del brazo. Empuje los husillos en las perillas correspondientes, ya montadas en el brazo 1, y fíjelos en su sitio ajustando los tornillos empotrados. Cada husillo debe posicionarse de modo que el borde plano quede afianzado mediante el tornillo empotrado

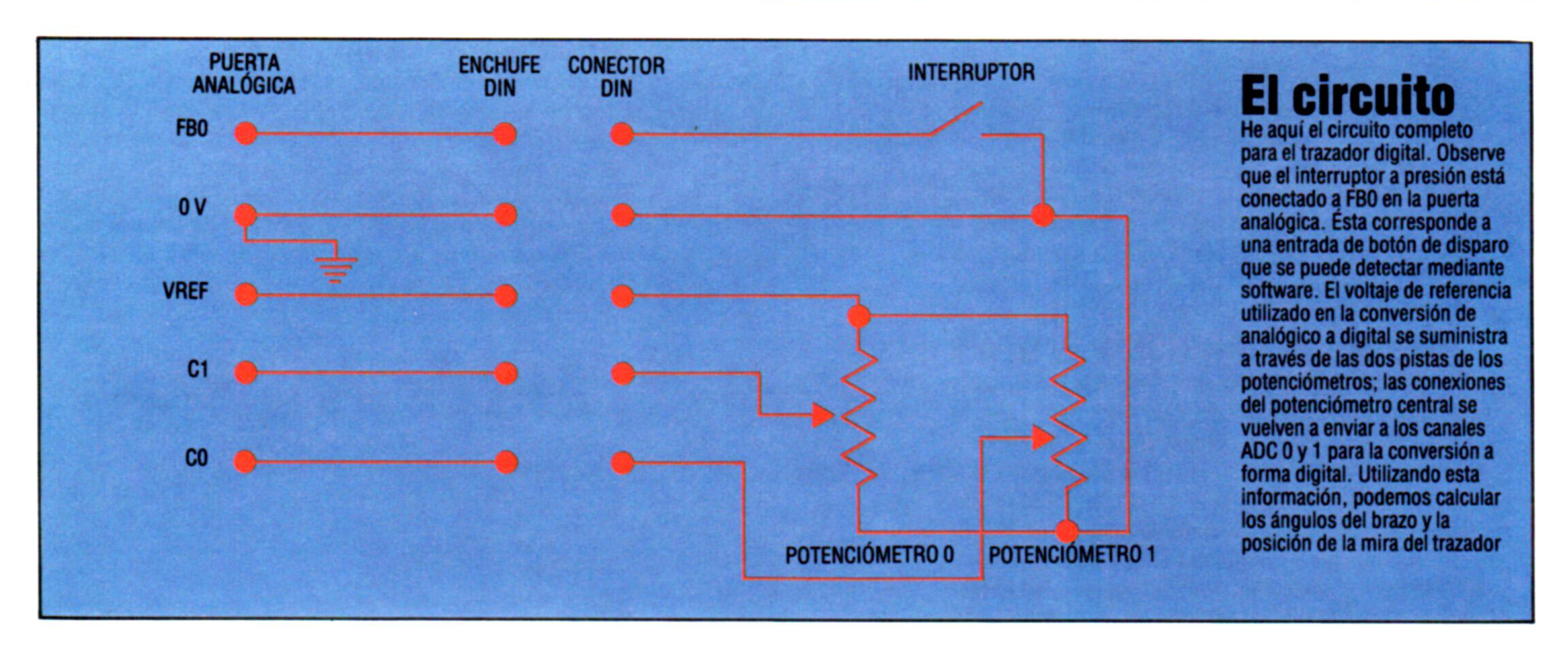
BRAZO 1
TUERCAS DEL POTENC.
BRAZO 2
POTENCIÓMETROS
CAJA

Paso 2: Montar la mira

Corte una mira de un trozo de plástico transparente relativamente delgado, como el que se utiliza para los estuches de cassettes de audio. La forma exacta de la mira no es importante, pero los ejes deben trazarse sobre la superficie de modo que se crucen (ver ilustración) y queden a 27 mm del centro del agujero de montaje. Fije la mira en el extremo libre del brazo 2 mediante el perno M5 y tres tuercas. Establezca la altura inicial en 32 mm. Esta disposición permitirá cambiar fácilmente la altura

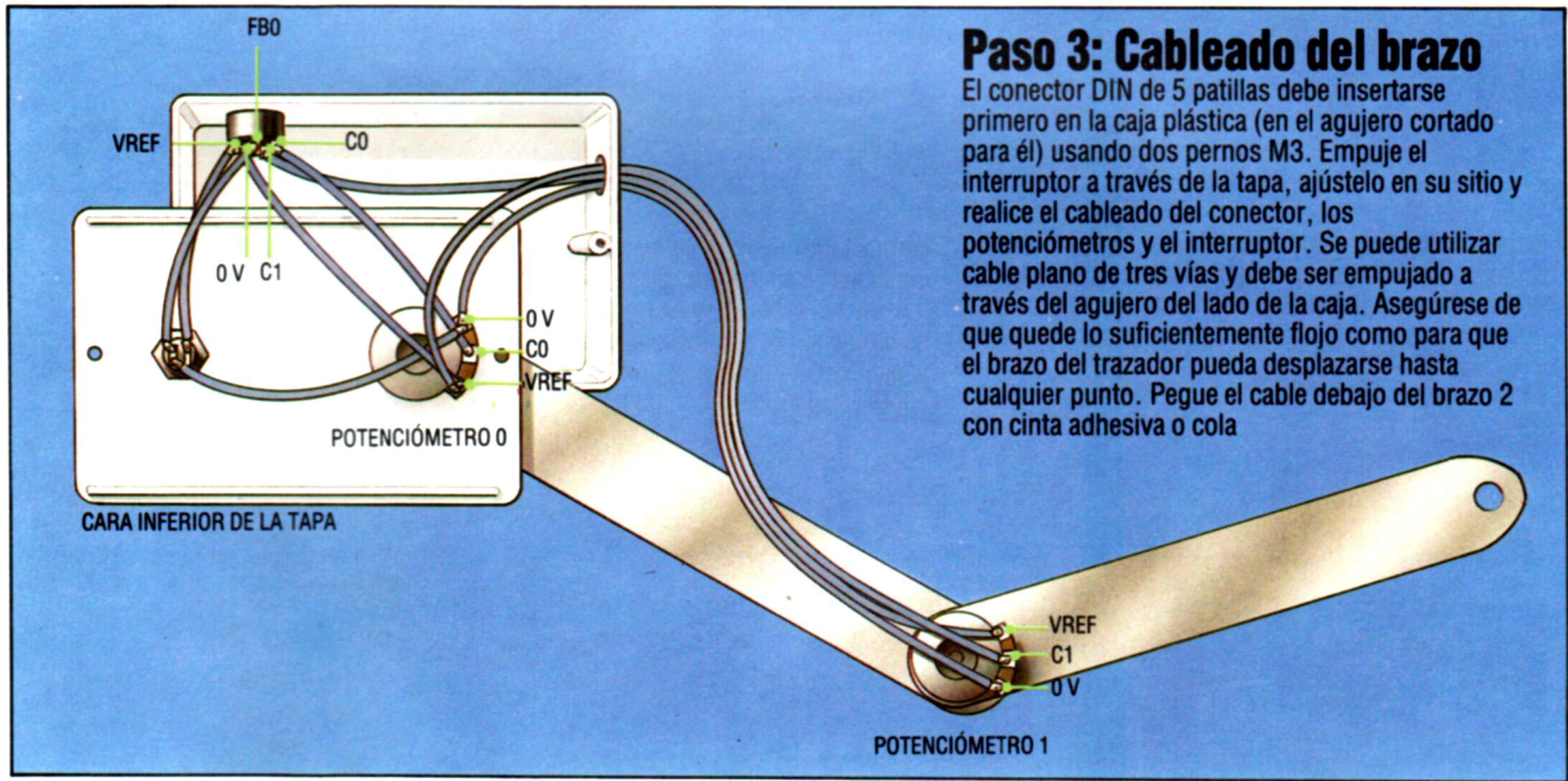






El circuito

He aquí el circuito completo para el trazador digital. Observe que el interruptor a presión está conectado a FB0 en la puerta analógica. Ésta corresponde a una entrada de botón de disparo que se puede detectar mediante software. El voltaje de referencia utilizado en la conversión de analógico a digital se suministra a través de las dos pistas de los potenciómetros; las conexiones del potenciómetro central se vuelven a enviar a los canales ADC 0 y 1 para la conversión a forma digital. Utilizando esta información, podemos calcular los ángulos del brazo y la posición de la mira del trazador

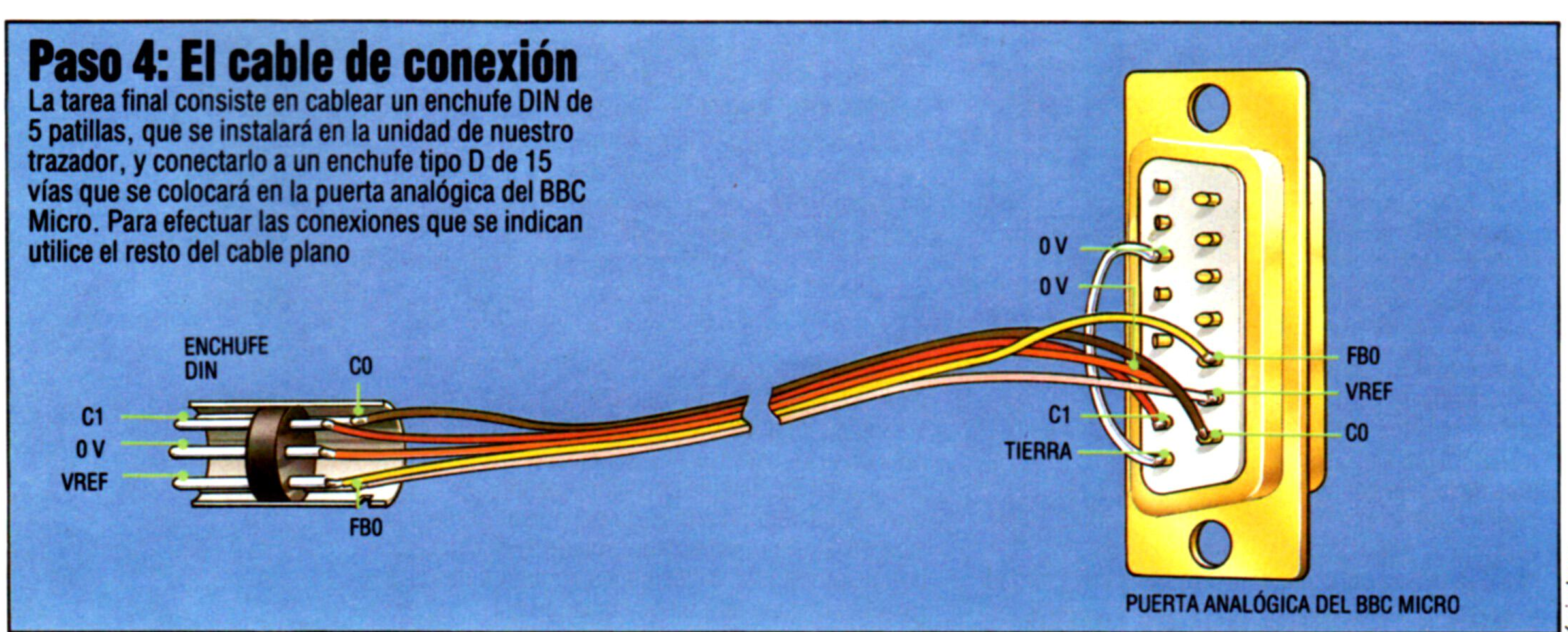


Paso 3: Cableado del brazo

El conector DIN de 5 patillas debe insertarse primero en la caja plástica (en el agujero cortado para él) usando dos pernos M3. Empuje el interruptor a través de la tapa, ajústelo en su sitio y realice el cableado del conector, los potenciómetros y el interruptor. Se puede utilizar cable plano de tres vías y debe ser empujado a través del agujero del lado de la caja. Asegúrese de que quede lo suficientemente flojo como para que el brazo del trazador pueda desplazarse hasta cualquier punto. Pegue el cable debajo del brazo 2 con cinta adhesiva o cola

Paso 4: El cable de conexión

La tarea final consiste en cablear un enchufe DIN de 5 patillas, que se instalará en la unidad de nuestro trazador, y conectarlo a un enchufe tipo D de 15 vías que se colocará en la puerta analógica del BBC Micro. Para efectuar las conexiones que se indican utilice el resto del cable plano



Kevin Jones

Revolución gráfica

Proseguimos nuestro estudio de los gráficos en 3 dimensiones del Commodore 64, analizando esta vez la conversión del listado en BASIC

El programa híbrido (*Prueba I-Rot* y *Hexa II-Rot*) anteriormente proporcionado se ejecuta con relativa rapidez. Pero es claro que el tener que inspeccionar la matriz de adyacencia E%(I,J) (que define los nodos que han de conectarse en la figura de líneas) con el solo objeto de determinar cuáles son los puntos que deben trazarse, ralentiza la marcha del programa.

Para darle mayor celeridad necesitamos codificar el resto del bucle clave en BASIC del programa *Cubo rotatorio* en código máquina. El resultado vale la pena por la velocidad ganada.

Para tratar los siguientes cálculos matemáticos:

```
X1%=X(I)+159:Y1%=199-(Z(I)+100)
X2%=X(J)+159:Y2%=199-(Z(J)+100)
```

que encontramos en las líneas 1640 y 1650 del programa original en BASIC, necesitamos otras llamadas al intérprete.

Esencialmente ambas líneas del BASIC toman una variable de coma flotante (p. ej., X(I)) y le añaden 159 (en coma flotante también) antes de aceptar la parte entera y almacenarla como X1% en dos bytes.

Las llamadas al intérprete necesarias para lograr esto son:

- FLPINT (dirección de la llamada $B1AA):
Esta rutina toma la parte entera del número en FAC y da el resultado (dentro del intervalo −32767 a 32767) en forma de byte *lo/hi* en los registros Y y A. Nótese aquí el orden *lo/hi* inusual, al revés que en muchas rutinas de intérprete.

- SNGFT (dirección de la llamada $B3A2):
Esta rutina toma un entero de un solo byte (un número entre 0 y 255) del registro Y, y lo coloca en FAC en coma flotante.

SNGFT se emplea en la subrutina ESTABLECIMIENTO del listado en assembly (línea 5150). Por ejemplo, el valor decimal 159 es colocado en el registro índice Y y se llama a SNGFT para convertirlo y situarlo en FAC.

Después de esto, MOVMF se encarga de poner el resultado en los cinco bytes de MEM1. Cuando deseamos sumar 159 sabemos que está disponible en MEM1.

Los problemas más importantes que surgen en la conversión a código máquina del bucle del BASIC se refieren a la especificación de determinados elementos en las tablas que definen la figura rotatoria. Puede que en algunos casos el cálculo de los punteros de la tabla resulte difícil. Las tablas de ordenadas X(I),Y(I),Z(I) no presentan dificultades especiales, ya que en cada tabla sencillamente añadimos cinco bytes al puntero para obtener la dirección del siguiente elemento.

La tabla E%(I,J), al ser bidimensional, es otra cosa.

Se ordena en la memoria secuencialmente de la siguiente manera:

```
E%(0,0)E%(1,0)E%(2,0)...E%(NP,0)
E%(0,1)E%(1,1)E%(2,1)...E%(NP,1), etc.
```

Es decir, la tabla se compone de bloques de memoria de 2×(NP+1) bytes de longitud cada uno, correspondiente a los valores del segundo subíndice, tomando cada elemento dos bytes (la tabla es de números enteros).

Nuestra intención es reflejar el código BASIC con la mayor exactitud en código máquina de tal modo que los bucles I,J (rastreadores de E%(I,J)) serán los siguientes:

```
FOR I=1 TO NP
FOR J=1 TO I
```

Esto significa que el cambio en el puntero para lograr un equivalente en código máquina de NEXT I, que permita incluir elementos con un primer subíndice cero, resulta algo complejo. Para hacer que la figura gire, se ha de acceder a los elementos de E%(I,J) en el siguiente orden:

```
E%(1,1)
E%(1,2)E%(2,2)
E%(1,3)E%(2,3)E%(3,3)
E%(1,4)E%(2,4)E%(3,4)E%(4,4), etc.
```

Un cálculo rápido muestra que hay que añadir 2×(NP+1) al puntero cada vez que se incrementa I. La mejor manera de hacer esto en el código máquina del 6502 es emplear el direccionamiento indirecto para acceder a E%(I,J).

El código sería:

```
LDY JINDEX
LDA (ZTEMP),Y
```

siendo ZTEMP la página cero, un puntero de dos bytes, y JINDEX sirve para seguir la pista de J. También ZTEMP debe ser incrementado a cada incremento de J. Los incrementos de ZTEMP y del registro Y se acompañan de un aumento del desplazamiento en dos bytes a cada incremento de J. El resultado final es que ZTEMP debe quedar incrementado en (2×NP+1)−(I−1) a cada iteración del bucle I.

La longitud del bloque es decrementada en (I−1) porque ZTEMP ha sido ya incrementada (I−1) veces en el bucle J ejecutado.

La expresión del cálculo del desplazamiento quiere decir que ZTEMP apunta al byte correcto después de ser incrementado I.

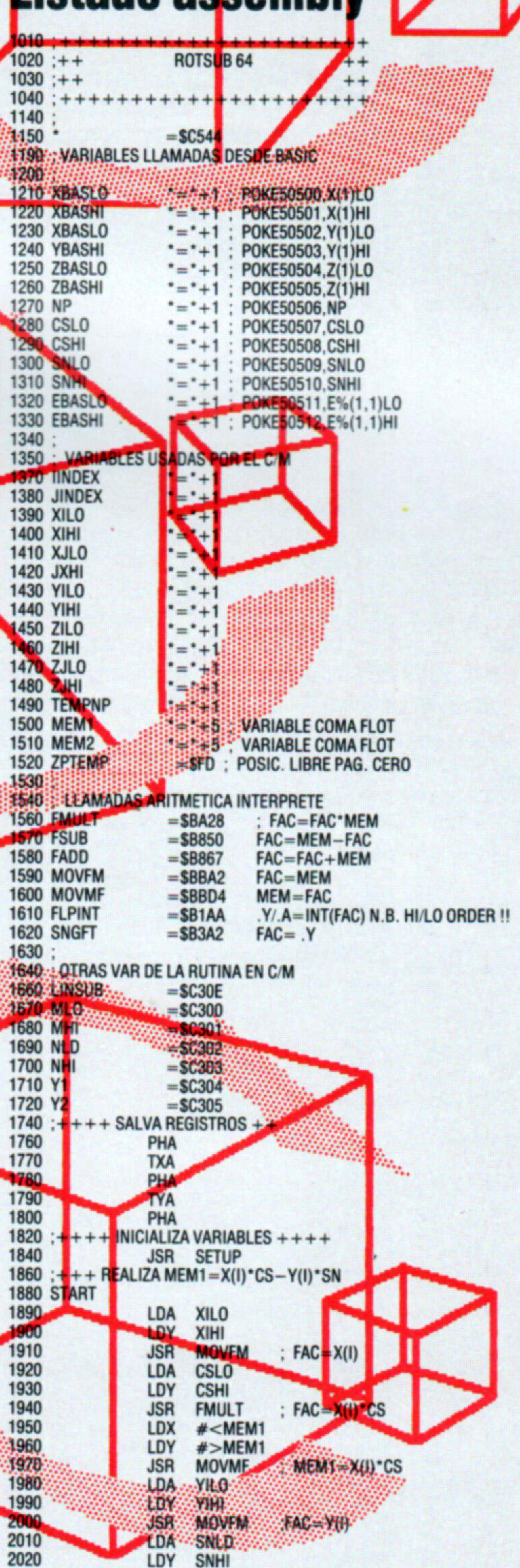
Por último, sería útil poder llamar a una rutina intérprete para que nos coloque la variable E%(I,J).

Tal rutina existe, pero resulta por desgracia tan sibilina (pues trata todo tipo de variables posibles) y tan lenta, que es preferible que calculemos nosotros mismos el desplazamiento de la dirección de E%(1,1).



Rutinas rotatorias

Los listados que siguen dan el código fuente del programa en código máquina que hace girar una figura de líneas, definida por la tabla E%(I,J) en una pantalla de alta resolución. El programa emplea las rutinas intérprete y métodos de posicionamiento de las variables del BASIC examinadas en el texto. Conseguimos ahora nuestro objetivo inicial de convertir el bucle original en BASIC que inspecciona las tablas para dar la figura rotatoria. Se incluyen además un cargador en BASIC para este código fuente, junto con un programa de prueba en BASIC que establece las variables en código máquina y llama a la rutina. El programa de prueba puede ser empleado con una versión ensamblada del código fuente, HEXA II-ROT o cargando y ejecutando el programa cargador, si pulsamos NEW y finalmente cargamos y ejecutamos el programa de prueba. En este segundo caso la línea 1030 será omitida

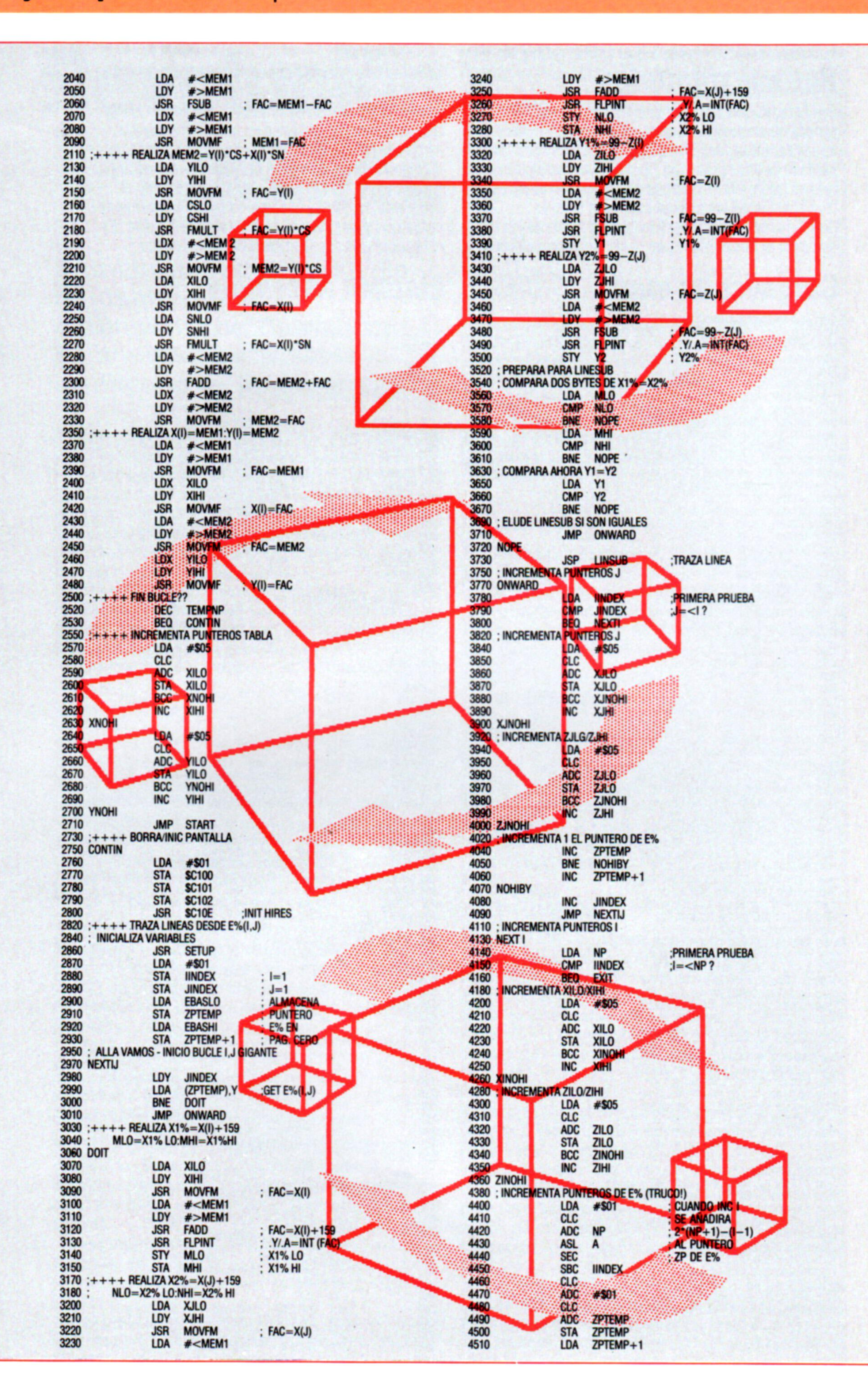


Cargador en BASIC

```
1000 REM *** INSERTAR HEXA II-ROT EN C/M ***
1010 REM ************************************
1020 DATA72,138,72,152,72,32,101,199
1030 DATA173,83,197,172,84,197,32,162
1040 DATA187,173,75,197,172,76,197,32
1050 DATA40,186,162,94,160,197,32,212
1060 DATA187,173,87,197,172,88,197,32
1070 DATA162,187,173,77,197,172,78,97
1080 DATA32,40,186,169,94,160,197,32,80
1090 DATA184,162,94,160,197,32,212,187
1100 DATA173,87,197,172,88,197,32,162
1110 DATA187,173,75,197,172,76,197,32
1120 DATA40,186,162,99,160,197,32,212
1130 DTA187,173,83,197,172,84,197,32
1140 DATA162,187,173,77,197,172,78,197
1150 DATA32,40,186,169,99,160,197,32
1160 DATA103,184,162,99,160,197,32,212
1170 DATA187,169,94,160,197,32,162,187
1180 DATA174,83,197,172,84,197,32,212
1190 DATA187,169,99,160,197,32,162,187
1200 DATA174,87,197,172,88,197,32,212
1210 DATA187,206,93,197,240,31,169,5,24
1220 DATA109,83,197,141,83,197,144,3
1230 DATA238,84,197,169,5,24,109,87,197
1240 DATA141,87,197,144,3,238,88,197,76
1250 DATA112,197,169,1,141,0,193,141,1
1260 DATA193,141,2,193,32,14,193,32,101
1270 DATA199,169,1,141,81,197,141,82
1280 DATA197,173,79,197,133,253,173,80
1290 DATA197,133,254,172,82,197,177,253
1300 DATA208,3,76,207,198,173,83,197
1310 DATA172,84,197,32,162,187,169,94
1320 DATA160,197,32,103,184,32,170,177
1330 DATA140,0,195,141,1,195,173,85,197
1340 DATA172,86,197,32,162,187,169,94
1350 DATA160,197,32,103,184,32,170,177
1360 DATA140,2,195,141,3,195,173,89,197
1370 DATA172,90,197,32,162,187,169,99
1380 DATA160,197,32,80,184,32,170,177
1390 DATA140,4,195,173,91,197,172,92
1400 DATA197,32,162,187,169,99,160,197
1410 DATA32,80,184,32,170,177,140,5,195
1420 DATA173,0,195,205,2,195,200,19,173
1430 DATA1,195,205,3,195,208,11,173,4
1440 DATA195,205,5,195,208,3,76,207,198
1450 DATA32,14,195,173,81,197,205,82
1460 DATA197,240,40,169,5,24,109,85,197
1470 DATA141,85,197,144,3,238,86,197
1480 DATA169,5,24,109,91,197,141,91,197
1490 DATA144,3,238,92,197,230,253,208,2
1500 DATA230,254,238,82,197,76,73,198
1510 DATA173,74,197,205,81,197,240,88
1520 DATA169,5,24,109,83,197,141,83,197
1530 DATA144,3,238,84,197,169,5,24,189
1540 DATA89,197,141,89,197,144,3,238,90
1550 DATA197,169,1,24,109,74,197,10,56
1560 DATA237,81,197,24,105,1,24,101,253
1570 DATA133,253,165,254,105,0,133,254
1580 DATA173,68,197,141,85,197,173,69
1590 DATA197,141,86,197,173,172,197,141
1600 DATA91,197,173,73,197,141,92,197
1610 DATA169,1,141,82,197,238,81,197,76
1620 DATA73,198,104,168,104,170,184,96
1630 DATA173,68,197,141,83,197,141,85
1640 DATA197,173,69,197,141,84,197,141
1650 DATA86,197,173,70,197,141,87,197
1660 DATA173,71,197,141,88,197,173,74
1670 DATA197,141,93,197,173,72,197,141
1680 DATA89,197,141,91,197,173,73,197
1690 DATA141,90,197,141,92,197,160,159
1700 DATA32,162,179,162,94,160,197,32
1710 DATA212,187,160,99,32,162,179,162
1720 DATA99,160,197,32,212,187,96
1730 DATA79160:REM*CHECKSUM*
1740 FORI=50536TO51123
1750 READX:POKEI,X:CC=CC+X
1760 NEXT
1770 READX:IFX<>CCTHENPRINT"ERROR EN SUMA CONTROL":STOP
1780 PRINT "HEXA II-ROT INSERTADO CORRECTAMENTE"
```

Listado assembly

```
1010 ;++++++++++++++++++++++++++
1020 ;++          ROTSUB 64        ++
1030 ;++                           ++
1040 ;++++++++++++++++++++++++++
1140 ;
1150 *             =$C544
1190 ; VARIABLES LLAMADAS DESDE BASIC
1200 ;
1210 XBASLO        *=*+1 ; POKE50500,X(1)LO
1220 XBASHI        *=*+1 ; POKE50501,X(1)HI
1230 XBASLO        *=*+1 ; POKE50502,Y(1)LO
1240 YBASHI        *=*+1 ; POKE50503,Y(1)HI
1250 ZBASLO        *=*+1 ; POKE50504,Z(1)LO
1260 ZBASHI        *=*+1 ; POKE50505,Z(1)HI
1270 NP            *=*+1 ; POKE50506,NP
1280 CSLO          *=*+1 ; POKE50507,CSLO
1290 CSHI          *=*+1 ; POKE50508,CSHI
1300 SNLO          *=*+1 ; POKE50509,SNLO
1310 SNHI          *=*+1 ; POKE50510,SNHI
1320 EBASLO        *=*+1 ; POKE50511,E%(1,1)LO
1330 EBASHI        *=*+1 ; POKE50512,E%(1,1)HI
1340 ;
1350 ; VARIABLES USADAS POR EL C/M
1370 IINDEX        *=*+1
1380 JINDEX        *=*+1
1390 XILO          *=*+1
1400 XIHI          *=*+1
1410 XJLO          *=*+1
1420 JXHI          *=*+1
1430 YILO          *=*+1
1440 YIHI          *=*+1
1450 ZILO          *=*+1
1460 ZIHI          *=*+1
1470 ZJLO          *=*+1
1480 ZJHI          *=*+1
1490 TEMPNP        *=*+1
1500 MEM1          *=*+5 ; VARIABLE COMA FLOT
1510 MEM2          *=*+5 ; VARIABLE COMA FLOT
1520 ZPTEMP        =$FD  ; POSIC. LIBRE PAG. CERO
1530 ;
1540 ; LLAMADAS ARITMETICA INTERPRETE
1560 FMULT         =$BA28   ; FAC=FAC*MEM
1570 FSUB          =$B850     FAC=MEM-FAC
1580 FADD          =$B867     FAC=FAC+MEM
1590 MOVFM         =$BBA2     FAC=MEM
1600 MOVMF         =$BBD4     MEM=FAC
1610 FLPINT        =$B1AA     .Y/.A=INT(FAC) N.B. HI/LO ORDER !!
1620 SNGFT         =$B3A2     FAC= .Y
1630 ;
1640 ; OTRAS VAR DE LA RUTINA EN C/M
1660 LINSUB        =$C30E
1670 MLO           =$C300
1680 MHI           =$C301
1690 NLD           =$C302
1700 NHI           =$C303
1710 Y1            =$C304
1720 Y2            =$C305
1740 ;++++ SALVA REGISTROS ++
1760               PHA
1770               TXA
1780               PHA
1790               TYA
1800               PHA
1820 ;++++ INICIALIZA VARIABLES ++++
1840               JSR   SETUP
1860 ;+++ REALIZA MEM1=X(I)*CS-Y(I)*SN
1880 START
1890               LDA   XILO
1900               LDY   XIHI
1910               JSR   MOVFM      ; FAC=X(I)
1920               LDA   CSLO
1930               LDY   CSHI
1940               JSR   FMULT      ; FAC=X(I)*CS
1950               LDX   #<MEM1
1960               LDY   #>MEM1
1970               JSR   MOVMF      ; MEM1=X(I)*CS
1980               LDA   YILO
1990               LDY   YIHI
2000               JSR   MOVFM      ;FAC=Y(I)
2010               LDA   SNLD
2020               LDY   SNHI
2030               JSR   FMULT      ; FAC=Y(I)*SN
```



```
2040          LDA   #<MEM1
2050          LDY   #>MEM1
2060          JSR   FSUB      ; FAC=MEM1-FAC
2070          LDX   #<MEM1
2080          LDY   #>MEM1
2090          JSR   MOVMF     ; MEM1=FAC
2110 ;++++ REALIZA MEM2=Y(I)*CS+X(I)*SN
2130          LDA   YILO
2140          LDY   YIHI
2150          JSR   MOVFM     ; FAC=Y(I)
2160          LDA   CSLO
2170          LDY   CSHI
2180          JSR   FMULT     ; FAC=Y(I)*CS
2190          LDX   #<MEM 2
2200          LDY   #>MEM 2
2210          JSR   MOVFM     ; MEM2=Y(I)*CS
2220          LDA   XILO
2230          LDY   XIHI
2240          JSR   MOVMF     ; FAC=X(I)
2250          LDA   SNLO
2260          LDY   SNHI
2270          JSR   FMULT     ; FAC=X(I)*SN
2280          LDA   #<MEM2
2290          LDY   #>MEM2
2300          JSR   FADD      ; FAC=MEM2+FAC
2310          LDX   #<MEM2
2320          LDY   #>MEM2
2330          JSR   MOVFM     ; MEM2=FAC
2350 ;++++ REALIZA X(I)=MEM1:Y(I)=MEM2
2370          LDA   #<MEM1
2380          LDY   #>MEM1
2390          JSR   MOVFM     ; FAC=MEM1
2400          LDX   XILO
2410          LDY   XIHI
2420          JSR   MOVMF     ; X(I)=FAC
2430          LDA   #<MEM2
2440          LDY   #>MEM2
2450          JSR   MOVFM     ; FAC=MEM2
2460          LDX   YILO
2470          LDY   YIHI
2480          JSR   MOVMF     ; Y(I)=FAC
2500 ;++++ FIN BUCLE??
2520          DEC   TEMPNP
2530          BEQ   CONTIN
2550 ;++++ INCREMENTA PUNTEROS TABLA
2570          LDA   #$05
2580          CLC
2590          ADC   XILO
2600          STA   XILO
2610          BCC   XNOHI
2620          INC   XIHI
2630 XNOHI
2640          LDA   #$05
2650          CLC
2660          ADC   YILO
2670          STA   YILO
2680          BCC   YNOHI
2690          INC   YIHI
2700 YNOHI
2710          JMP   START
2730 ;++++ BORRA/INIC PANTALLA
2750 CONTIN
2760          LDA   #$01
2770          STA   $C100
2780          STA   $C101
2790          STA   $C102
2800          JSR   $C10E     ;INIT HIRES
2820 ;++++ TRAZA LINEAS DESDE E%(I,J)
2840 ;  INICIALIZA VARIABLES
2860          JSR   SETUP
2870          LDA   #$01
2880          STA   IINDEX        ; I=1
2890          STA   JINDEX        ; J=1
2900          LDA   EBASLO        ; ALMACENA
2910          STA   ZPTEMP        ; PUNTERO
2920          LDA   EBASHI        ; E% EN
2930          STA   ZPTEMP+1      ; PAG. CERO
2950 ;  ALLA VAMOS - INICIO BUCLE I,J GIGANTE
2970 NEXTIJ
2980          LDY   JINDEX
2990          LDA   (ZPTEMP),Y    ;GET E%(I,J)
3000          BNE   DOIT
3010          JMP   ONWARD
3030 ;++++ REALIZA X1%=X(I)+159
3040 ;     MLO=X1% LO:MHI=X1%HI
3060 DOIT
3070          LDA   XILO
3080          LDY   XIHI
3090          JSR   MOVFM         ; FAC=X(I)
3100          LDA   #<MEM1
3110          LDY   #>MEM1
3120          JSR   FADD          ; FAC=X(I)+159
3130          JSR   FLPINT        ; .Y/.A=INT (FAC)
3140          STY   MLO           ; X1% LO
3150          STA   MHI           ; X1% HI
3170 ;++++ REALIZA X2%=X(J)+159
3180 ;     NLO=X2% LO:NHI=X2% HI
3200          LDA   XJLO
3210          LDY   XJHI
3220          JSR   MOVFM         ; FAC=X(J)
3230          LDA   #<MEM1
3240          LDY   #>MEM1
3250          JSR   FADD          ; FAC=X(J)+159
3260          JSR   FLPINT        ; .Y/.A=INT(FAC)
3270          STY   NLO           ; X2% LO
3280          STA   NHI           ; X2% HI
3300 ;++++ REALIZA Y1%=99-Z(I)
3320          LDA   ZILO
3330          LDY   ZIHI
3340          JSR   MOVFM         ; FAC=Z(I)
3350          LDA   #<MEM2
3360          LDY   #>MEM2
3370          JSR   FSUB          ; FAC=99-Z(I)
3380          JSR   FLPINT        ; .Y/.A=INT(FAC)
3390          STY   Y1            ; Y1%
3410 ;++++ REALIZA Y2%=99-Z(J)
3430          LDA   ZJLO
3440          LDY   ZJHI
3450          JSR   MOVFM         ; FAC=Z(J)
3460          LDA   #<MEM2
3470          LDY   #>MEM2
3480          JSR   FSUB          ; FAC=99-Z(J)
3490          JSR   FLPINT        ; .Y/.A=INT(FAC)
3500          STY   Y2            ; Y2%
3520 ; PREPARA PARA LINESUB
3540 ; COMPARA DOS BYTES DE X1%=X2%
3560          LDA   MLO
3570          CMP   NLO
3580          BNE   NOPE
3590          LDA   MHI
3600          CMP   NHI
3610          BNE   NOPE
3630 ; COMPARA AHORA Y1=Y2
3650          LDA   Y1
3660          CMP   Y2
3670          BNE   NOPE
3690 ; ELUDE LINESUB SI SON IGUALES
3710          JMP   ONWARD
3720 NOPE
3730          JSP   LINSUB        ;TRAZA LINEA
3750 ; INCREMENTA PUNTEROS J
3770 ONWARD
3780          LDA   IINDEX        ;PRIMERA PRUEBA
3790          CMP   JINDEX        ;J=<I ?
3800          BEQ   NEXTI
3820 ; INCREMENTA PUNTEROS J
3840          LDA   #$05
3850          CLC
3860          ADC   XJLO
3870          STA   XJLO
3880          BCC   XJNOHI
3890          INC   XJHI
3900 XJNOHI
3920 ; INCREMENTA ZJLG/ZJHI
3940          LDA   #$05
3950          CLC
3960          ADC   ZJLO
3970          STA   ZJLO
3980          BCC   ZJNOHI
3990          INC   ZJHI
4000 ZJNOHI
4020 ; INCREMENTA 1 EL PUNTERO DE E%
4040          INC   ZPTEMP
4050          BNE   NOHIBY
4060          INC   ZPTEMP+1
4070 NOHIBY
4080          INC   JINDEX
4090          JMP   NEXTIJ
4110 ; INCREMENTA PUNTEROS I
4130 NEXT I
4140          LDA   NP            ;PRIMERA PRUEBA
4150          CMP   IINDEX        ;I=<NP ?
4160          BEQ   EXIT
4180 ; INCREMENTA XILO/XIHI
4200          LDA   #$05
4210          CLC
4220          ADC   XILO
4230          STA   XILO
4240          BCC   XINOHI
4250          INC   XIHI
4260 XINOHI
4280 ; INCREMENTA ZILO/ZIHI
4300          LDA   #$05
4310          CLC
4320          ADC   ZILO
4330          STA   ZILO
4340          BCC   ZINOHI
4350          INC   ZIHI
4360 ZINOHI
4380 ; INCREMENTA PUNTEROS DE E% (TRUCO!)
4400          LDA   #$01          ; CUANDO INC I
4410          CLC                 ; SE AÑADIRA
4420          ADC   NP            ; 2*(NP+1)-(I-1)
4430          ASL   A             ; AL PUNTERO
4440          SEC                 ; ZP DE E%
4450          SBC   IINDEX
4460          CLC
4470          ADC   #$01
4480          CLC
4490          ADC   ZPTEMP
4500          STA   ZPTEMP
4510          LDA   ZPTEMP+1
```

```
4520             ADC   #$00
4530             STA   ZPTEMP+1
4550 ; REINICIALIZA XJLO/XJHI
4570             LDA   XBASLO
4580             STA   XJLO
4590             LDA   XBASHI
4600             STA   XJHI
4620 ; REINICIALIZA ZJLO/ZJHI
4640             LDA   ZBASLO
4650             STA   ZJLO
4660             LDA   ZBASHI
4670             STA   ZJHI
4690 ; REINICIALIZA JINDEX
4710             LDA   #$01
4720             STA   JINDEX
4740 ; INCREMENTA IINDEX
4760             INC   IINDEX
4770             JMP   NEXTIJ
4790 ; EL BUCLE GIGANTE ACABA AQUI
4800 ;
4810 ;++++ RESTAURA LOS REGISTROS ++++
4820 ;
4830 EXIT
4840             PLA
4850             TAY
4860             PLA
4870             TAX
4880             PLA
4890             RTS
4930 ;++++ SUBRUTINA
4940 ;
4950 SETUP
4960             LDA   XBASLO
4970             STA   XILO
4980             STA   XJLO
4990             LDA   XBASHI
5000             STA   XIHI
5010             STA   XJHI
5020             LDA   YBASLO
5030             STA   YILO
5040             LDA   YBASHI
5050             STA   YIHI
5060             LDA   NP
5070             STA   TEMPNP
5080             LDA   ZBASLO
5090             STA   ZILO
5100             STA   ZJLO
5110             LDA   ZBASHI
5120             STA   ZIHI
5130             STA   ZJHI
5140             LDY   #159
5150             JSR   SNGFT        ; FAC=159
5160             LDX   #<MEM1
5170             LDY   #>MEM1
5180             JSR   MOVMF        ; MEM1=159 IN FPT
5190             LDY   #99
5200             JSR   SNGFT        ; FAC=99
5210             LDX   #<MEM2
5220             LDY   #>MEM2
5230             JSR   MOVMF        ; MEM2=99 IN FTP
5240             RTS
```

Programa de prueba en BASIC

```
1000 REM ** TRAZADO CUBOS ROTATORIOS **
1010 IFA=0THENA=1:LOAD"PLOTSUB.HEX",8,1
1020 IFA=1THENA=2:LOAD"LINESUB.HEX",8,1
1030 IFA=2THENA=3:LOAD"II-ROT.HEX",8,1
1040 PRINTCHR$(147)"ESPERA 17 SEGUNDOS"
1050 REM**DIMENSIONA LAS TABLAS**
1060 NP=24:REM NUMERO DE PUNTOS
1070 DIM X(NP),Y(NP),Z(NP)
1080 DIM E%(NP,NP):REM CONEXION DE LAS ESQUINAS
1090 REM**INICIALIZA TABLAS**
1100 REM--DATOS INICIALES COORDENADAS CUBO
1110 DATA   50,  50,  50:REM----------/1
1120 DATA  -50,  50,  50:REM TOP FOUR /2
1130 DATA  -50, -50,  50:REM POINTS /3
1140 DATA   50, -50,  50:REM----------/4
1150 DATA   50,  50, -50:REM----------/5
1160 DATA  -50,  50, -50:REM BOT FOUR /6
1170 DATA  -50, -50, -50:REM POINTS /7
1180 DATA   50, -50, -50:REM----------/8
```

La siguiente sección del programa lee (READ) los datos definidos anteriormente en las tablas X(), Y(), Z(). Para cubos minúsculos los valores DATA iniciales se reducirán mediante el coef. 0.3

```
1190 REM ** LEE DATOS DE CUBOS 3-D
1200 FORI=1TO8:REM CENTRO DEL CUBO GRANDE
1210 READX(I),Y(I),Z(I)
1220 NEXT
1230 RESTORE
1240 FORI=9TO16:REM CUBO PEQUEÑO DE LA DERECHA
1250 READX,Y,Z
1260 X(I)=.3*X+120:Y(I)=.3*Y:Z(I)=.3*Z
1270 NEXT
1280 RESTORE
1290 FORI=17TO24:REM CUBO PEQUEÑO IZQ
1300 READX,Y,Z
1310 X(I)=.3*X-120:Y(I)=.3*Y:Z(I)=.3*Z
1320 NEXT
```

La siguiente sección hace girar las coordenadas de los puntos equivalente a una perspectiva cambiada en 45°

```
1330 REM ** ROTACION ESPACIAL SOBRE EJE X **
1340 FORI=1TONP
1350 Y(I)=Y(I)*COS(π/4)-Z(I)*SIN(π/4)
1360 Z(I)=Z(I)*COS(π/4)+Y(I)*SIN(π/4)
1370 NEXT
1380 REM** ROTACION ESPACIAL DE π/4 SOBRE EL EJE Z
1390 FORI=1TONP
1400 X(I)=X(I)*COS(π/4)-Y(I)*SIN(π/4)
1410 Y(I)=Y(I)*COS(π/4)+X(I)*SIN(π/4)
1420 NEXT
```

La siguiente sección define los puntos que se han de conectar dentro de la figura en cada uno de los tres cubos, mediante la tabla de adyacencia E%(,)

```
1430 REM ** DATOS CONEXION ESQUINAS **
1440 REM--CUBO TAMAÑO INTERMEDIO
1450 E%(1,2)=1:REM CONEXION DE 1 CON 2
1460 E%(2,3)=1:E=(3,4)=1:E%(4,1)=1
1470 E%(5,6)=1:REM CUADRADO BASE
1480 E%(6,7)=1:E%(7,8)=1:E%(8,5)=1
1490 E%(5,1)=1:REM ESQUINAS DE ARRIBA ABAJO
1500 E%(6,2)=1:E%(7,3)=1:E%(8,4)=1
1510 REM-------------------------
1520 REM—CUBO PEQUEÑO DERECHA
1530 FORI=9TO16:FORJ=9TO16
1540 E%(I,J)=E%(I-8,J-8)
1550 NEXT:NEXT
1560 REM-LEFT SMALL CUBE
1570 FORI=17TO24:FORJ=17TO24
1580 E%(I,J)=E%(I-8,J-8)
1590 NEXT:NEXT
1600 REM**SIMETRIZA E%(I,J)**
1610 FORI=1TONP:FORJ=1TONP
1620 IFE%(I,J)<>0THENE%(J,I)=1
1630 NEXT:NEXT
```

En este punto el programa llama al código máquina para que ejecute los cálculos necesarios del giro y dibuje la nueva figura. Esta llamada se hace reiteradamente desde el interior de un bucle que hace girar los tres cubos 360° antes de terminar

```
1640 REM ****************************************
1650 REM ** TRAZADO CUBO ROTATORIO **
1660 SA=2*π/45:CS=COS(SA):SN=SIN(SA)
1670 GOSUB1790:REM INICIALIZACION
1680 FOR A=0 TO 2*π STEP SA
1690 SYS50536:REM ROTACION
1700 NEXTA:REM ANGULO SIGUIENTE
1710 GETA$:IFA$=""THEN1710
1720 REM ####################################
1730 GOSUB1750:REM RESTAURA PANTALLA
1740 END
1750 REM**RESTAURA PANTALLA**
1760 POKE49408,0:SYS49422
1770 PRINTCHR$(147)
1780 RETURN
1790 REM**INICIALIZA ROTSUB**
1800 REM*****************************************
1810 REM**          NOTA—NO DEFINIR NUEVAS        **
1820 REM**           VARIABLES ENTRE ESTA         **
1830 REM**          SUBRUTINA Y LA LLAMADA        **
1840 REM**              AL C/M YA QUE             **
1850 REM**          CAMBIARIAN LAS TABLAS         **
1860 REM**            DE DIRECCION BASE           **
1870 REM*****************************************
1880 X(1)=X(1):REM X(1) VARIABLE EN CURSO
1890 POKE50500,PEEK(71):REM X(1)LO
1900 POKE50501,PEEK(72):REM X(1)HI
1910 Y(1)=Y(1):REM Y(1) VARIABLE EN CURSO
1920 POKE50502,PEEK(71):REM Y(1)LO
1930 POKE50503,PEEK(72):REM Y(1)HI
1940 Z(1)=Z(1):REM Z(1) VARIABLE EN CURSO
1950 POKE50504,PEEK(71)
1960 POKE50505,PEEK(72)
1970 POKE50506,NP:REM NUMERO DE PUNTOS
1980 CS=CS:REM HACE A CS VAR EN CURSO
1990 POKE50507,PEEK(71)
2000 POKE50508,PEEK(72)
2010 SN=SN:REM HACE A SN VAR EN CURSO
2020 POKE50509,PEEK(71)
2030 POKE50510,PEEK(72)
2040 E%(1,1)=E%(1,1):REM E% VARIABLE EN CURSO
2050 POKE50511,PEEK(71)
2060 POKE50512,PEEK(72)
2070 RETURN
```

Mano a mano

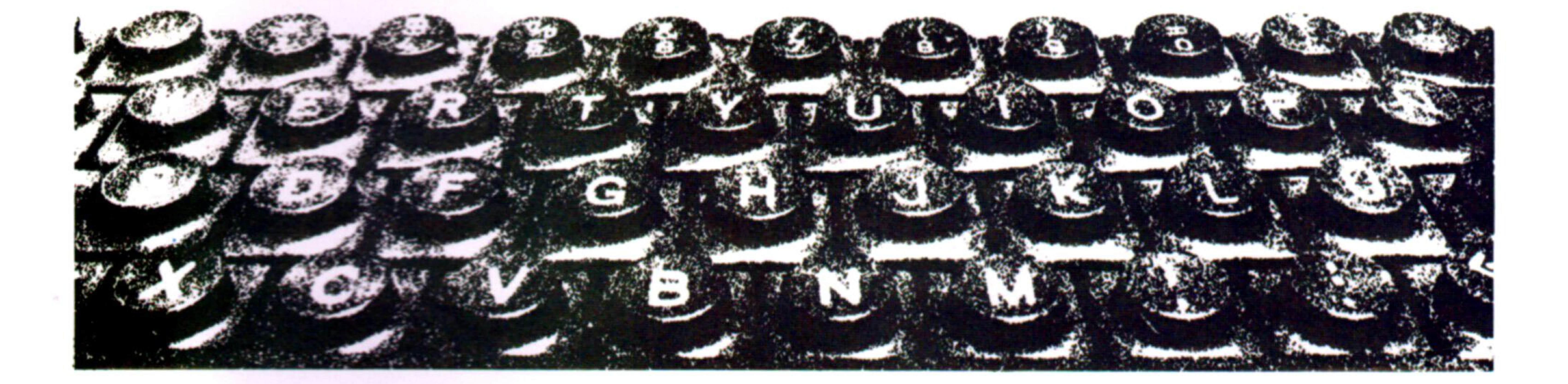
He aquí un interesante juego, conocido bajo la denominación "Piedra, papel y tijeras". Esta versión en BASIC ha sido escrita por Peter Shaw para el Oric



Tradicionalmente este juego se practica con las manos. Los dos jugadores, con sus manos a la espalda, deben mostrarlas simultáneamente imitando con ellas el objeto escogido. Deben elegir entre piedra, papel y tijeras. Las tijeras cortan el papel, la piedra rompe las tijeras, el papel cubre la piedra. Una vez hecha la elección, pulse C para tijeras, R para piedra, P para papel. Gana el jugador que totalice diez puntos.

```
10 REM PIEDRA, PAPEL, TIJERAS
20 CLS**PETER SHAW**
30 PAPER 0:INK 3
40 A=INT(RND(1)*3):PING
50 PLOT 12,2,"PIEDRA-PAPEL-TIJERAS"
55 PLOT 6,4,"HUMANO  "+STR$(HS)
56 PLOT 29,4,"YO  "+STR$(OS)
60 GET A$:ZAP:FOR P=0 TO 7:INK P:WAIT 4;
   NEXT P
65 CLS
70 IF A$="R" THEN V=0
80 IF A$="P" THEN V=1
90 IF A$="C" THEN V=2
100 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:
    PRINT
110 PRINT"YO TENGO ";
120 IF A=0 THEN PRINT"PIEDRA"
130 IF A=1 THEN PRINT"PAPEL"
140 IF A=2 THEN PRINT"TIJERAS"
150 PRINT:PRINT:PRINT
160 PRINT"TU TIENES  ";
170 IF V=0 THEN PRINT"PIEDRA"
180 IF V=1 THEN PRINT"PAPEL"
190 IF V=2 THEN PRINT"TIJERAS"
200 PRINT:PRINT:PRINT:SHOOT
210 IF V=A THEN PRINT"EMPATE"
220 IF V=0 AND A=1 THEN PRINT"PAPEL
    CUBRE PIEDRA. YO GANO":OS=OS+1
230 IF V=0 AND A=2 THEN PRINT"PIEDRA ROMPE
    TIJERAS. TU GANAS":HS=
    HS+1
240 IF V=1 AND A=0 THEN PRINT"PAPEL
    CUBRE PIEDRA. TU GANAS":HS=
    HS+1
250 IF V=1 AND A=2 THEN PRINT"TIJERAS
    CORTAN PAPEL. TU GANAS":OS=OS+1
260 IF V=2 AND A=0 THEN PRINT "PIEDRA ROMPE
    TIJERAS. YO GANO":OS=OS+1
270 IF V=2 AND A=1 THEN PRINT"TIJERAS
    CORTAN PAPEL. TU GANAS":HS=HS+1
280 PLOT 13,23,"PULSA UNA TECLA"
290 EXPLODE:WAIT 30
295 IF HS=10 OR OS=10 THEN 310
300 GOTO 30
310 CLS
320 IF HS=10 THEN PING:PRINT"BRAVO.TU
    GANAS"
330 IF OS=10 THEN ZAP:PRINT"YO GANO OTRA
    VEZ"
340 PRINT:PRINT:PRINT
345 WAIT 20:EXPLODE
350 END
```

Editorial Delta, S.A.



10088

9 788485 822836